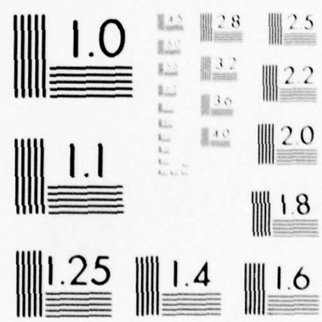
AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955
UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

1 OF 6
AD
A070973



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Report No. FAA-RD-79-33,I

LEVEL III

AD A070973

CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT

Volume I - Operational Software Complex

DDC RECEIVED JUL 10 1979

January 1979

Final Report

DDC FILE COPY

Document is available to the U.S. public through the National Technical Information Service, Springfield, Virginia 22161.

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
Systems Research & Development Service
Washington, D.C. 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

Technical Report Documentation Page

1. Report No. 19	2. Government Accession No.	3. Recipient's Catalog No.
18 FAA-RD-79-33-1		

4. Title and Subtitle
6 Central Flow Control
Software Design Document, Volume I.
Operational Software Complex (OPCX).

5. Report Date
11 Jan~~uary~~ 1979

6. Performing Organization Code

7. Author(s)
Computer ~~Sciences~~ Software Corporation

8. Performing Organization Report No.
14 CSC/SD-78/6172-1

9. Performing Organization Name and Address
Computer Sciences Corporation
System Sciences Corporation
8728 Colesville Road
Silver Spring, Maryland 20910

10. Work Unit No. (TRAIS)

11. Contract or Grant No.
15 DOT-FA77WA-3955

13. Type of Report and Period Covered
9 Final Report.

12. Sponsoring Agency Name and Address
U. S. Department of Transportation
Federal Aviation Administration
Systems Research and Development Service
Washington, D.C. 20591

14. Sponsoring Agency Code
ARD-102

15. Supplementary Notes
12 543p.
Vol 2 - A070772

16. Abstract

The Software Design Document (SDD), ~~a deliverable under Contract DOT-FA77WA-3955,~~ is a design document that defines the translation of the Central Flow Control (CFC) Software System functional requirements established in the Federal Aviation Administration's (FAA) Computer Program Functional Specifications (CPFSs) into implemented Software Programs.

The SDD describes the exact configuration of the computer programs produced for the CFC Software System. It provides a complete technical description of the software functions, structures, operating environment, data organization, visual table of contents, program listings, and data and module cross references.

Volume I describes the Operational Complex (OPCX) which operates in real-time mode. The OPCX is comprised of the Executive, Database, Simulation, and Applications subsystems.

17. Key Words
CENTRAL FLOW CONTROL
REAL-TIME PROGRAM
PRODUCT SPECIFICATION

18. Distribution Statement
This document is available to the public through the National Technical Information Service (NTIS), Springfield, Virginia 22151.

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	541	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

408 479

mt

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price \$2.25, SD Catalog No. C13.10:286.

Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F

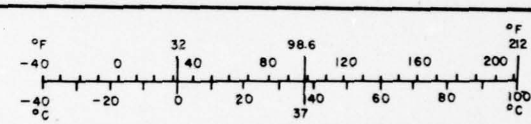


TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Avail and/or special
Dist
A

LIST OF ILLUSTRATIONS

VOLUME 1

VOLUME 2

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

SECTION 1 - INTRODUCTION

The Central Flow Control (CFC) Software Design Document (SDD) translates the functional requirements established in the Computer Program Functional Specifications (CPFSs) into physical software implementations. The SDD describes the structure of the CFC Software System, the decomposition of the System into its major parts, the function of each part, the precise interface among the parts, and the implementation details for each software element.

Section 1.1 summarizes the circumstances and events leading to the development of this SDD. It contains a general description of the operational problem to be solved, and it references the CPFSs, the Program Design Specification(PDS) and studies that have a significant bearing on the design approach. Section 1.2 describes the purpose and scope of the SDD and indicates its relationship to the CPFSs and the PDS. Section 1.3 lists the documents either cited or used as basic references in the SDD and cites the contract as the authority for the preparation of the SDD. Section 1.4 contains definitions of terms used to describe the hierarchy of software elements in the System. It also contains definitions of Committee on Data Systems Languages (CODSAYL) terms used to define the hierarchy of data-structure elements.

1.1 BACKGROUND

The Air Traffic Control System Command Center (ATCSCC) was established in 1970 to oversee the flow of aircraft among Air Route Traffic Control Centers (ARTCCs). The ATCSCC's primary objective is to balance national air traffic

flow to minimize delays equitably without exceeding controller capacity or jeopardizing safety.

The ATCSCC consists of the following functional elements:

- Central Flow Control Function (CFCF)
- Airport Reservation Function (ARF)
- Central Altitude Reservation Function (CARF)
- Contingency Command Post (CCP)

The CFCF has had automation support for some time; the CFC Software System will provide further automation. In January 1972, a version of the automation program, the Airport Information Retrieval System (AIRS), was placed in operation by the Transportation Systems Center using a commercial timesharing computing system. AIRS provides terminal arrival delay predictions, flow rated for quota flow procedures, and fuel advisory departure (FAD) procedure support. The primary data source for the System is the Official Airline Guide (OAG) published by the R. H. Donnelley Corporation. AIRS uses a Digital Equipment Corporation (DEC) System 10 timesharing computer with associated communications interfaces and input/output devices at the ATCSCC.

In early 1976, an FAA design team produced a software design and a new set of CPFSs for the CFC Software System. These documents were used as the basis for a Request for Proposal (RFP) for development of the CFC Software System. After a competitive procurement, Computer Sciences Corporation (CSC) was awarded the development contract in April 1977. Under contract, CSC developed the CFC Program Design Specification (PDS), which was approved by the FAA Critical Design Review, and developed the CFC System in accordance with

that design. The CFC System will improve the automation support provided to the ATCSCC in Washington, D.C. A major area of improvement will be increased accuracy in simulation and estimation, resulting from the addition of real-time inputs to the information in the OAG data base. The CFC System has been implemented on a dedicated International Business Machines (IBM) 9020A computer system located at the ARTCC in Hilliard, Florida, and digitally interfaced with the 20 ARTCCs located throughout the continental United States.

The design in this SDD is based on the CPFSs (References 1 through 7) and the PDS (Reference 8).

1.2 SCOPE AND PURPOSE

1.2.1 Purpose

The purpose of the SDD is to document the design of the CFC Software System. Formulated from the CPFS and the PDS, the SDD presents the detailed design as represented by the resulting CFC System.

1.2.2 Scope

The SDD describes the structure of the System, the decomposition of the System into its major parts, the function of each part, and the precise interface among the parts.

The SDD contains four sections. Section 1, Introduction, was summarized above. Section 2, System Architecture, provides the documentation of the mapping of the logical requirements of the CPFS into the physical elements of software as implemented. Section 3, System Data, provides the documentation of the

mapping of the logical data requirements of the CPFS into the physical data components of the implementation. Section 4, System Programs, provides the documentation of the mapping of the logical processing requirements of the CPFS into the modules (units of software) of the implementation.

Numerous documentation techniques have been employed to facilitate reader comprehensability and document updating. For example, to describe the System architecture, in Section 2, hierarchy diagrams and thread diagrams have been used. Hierarchy diagrams provide (1) a visual table of contents of the hierarchy of the software architecture, and (2) for each element of the hierarchy, a narrative description of the function performed by that element. Thread diagrams show the sequence of functions that will be performed based on a particular set of input stimuli.

To facilitate updating of detailed information, certain portions of the CFC SDD are computer-generated. In Section 3, the data item cross-references are produced by the Automated Code Auditor (MA).

In Section 4, the Program Hierarchy, Program Descriptor Blocks, and Program Cross References are produced by the JOVIAL Automated Verification System (JA). Also to facilitate document maintainability, the data item descriptions (Section 3.4) and the module engineering notebooks are included by reference, since these are subject to change periodically. The data item descriptions consist of commented COMPOOL listings and the module engineering notebooks contain the module prologue, PDL, test specifications, and commented source code. The module prologues and PDL are in the form of block comments and are maintained in the same library as the associated source code.

1.3 REFERENCES AND AUTHORITY

1.3.1 References

The following documents are either cited or used as basic references in the SDD. When one of the following documents is referenced in this SDD, a superscript with the reference number in parentheses is used. When this is not appropriate or confusing, the phrase Reference "i" is used where i is the reference number.

The Computer Program Functional Specifications consist of References 1 through 7. The Program Design Specification (PDS) is Reference 8.

1. U.S. Department of Transportation, Report No. FAA-RD-76-157,I, Central Flow Control Computer Program Specifications: Volume I, Introduction to Specification Series, System Overview, Central Flow Control Design Team, Federal Aviation Administration, September 1976, Final Report

2. U.S. Department of Transportation, Report No. FAA-RD-76-157,II, Central Flow Control Computer Program Specifications: Volume II, Application Program Specification, Central Flow Control Design Team, Federal Aviation Administration, September 1976, Final Report

3. U.S. Department of Transportation, Report No. FAA-RD-76-157, III, Central Flow Control Computer Program Specifications: Volume III, Off-Line Support Subsystem Specification, Central Flow Control Design Team, Federal Aviation Administration, September 1976, Final Report

4. U.S. Department of Transportation, Report No. FAA-RD-76-157, IV, Central Flow Control Computer Program Specifications: Volume IV, Data Base Subsystem Specification, Central Flow Control Design Team, Federal Aviation Administration, September 1976, Final Report

5. U.S. Department of Transportation, Report No. FAA-RD-76-157, V, Central Flow Control Computer Program Specifications: Volume V, Executive Subsystem Specification, Central Flow Control Design Team, Federal Aviation Administration, September 1976, Final Report

6. U.S. Department of Transportation, Report No. FAA-RD-76-157, II, Central Flow Control Software Systems Analysis, Volume II, Data Base Software Study, Kershaw, George G., and Guye, Travis D.; DBM Corporation, July 1976, Final Report

7. U.S. Department of Transportation, Report No. FAA-RD-76-144, A Computer Program Functional Design of the Simulation Subsystem of an Automated Central Flow Control System, Medeiros, Manual F.; MacDonald, Paul M.; Maglione, Vito P.; and Wright, Richard D., U. S. Department of Transportation, Transportation Systems Center, August 1976, Final Report

8. Computer Sciences Corporation, Report No. CSC/SD-77/6093, Central Flow Control Program Design Specifications, prepared for U.S. Department of Transportation, Federal Aviation Administration, Systems Research & Development Service, August 1977, Contract No. DOT-FAA77WA-3955

1.3.2 Authority

The authority for the preparation of the SDD is the Contract Schedule.

1.4 DEFINITIONS

This section provides definitions of: (1) terms used to describe the elements of software in the CFC Software System hierarchy, (2) CODASYL terms used to define the data hierarchy used in System documentation, and (3) the glossary of mnemonics, acronyms, phrases, and other commonly used CFC terms.

1.4.1 Elements of the CFC Software System Hierarchy.

The elements of the CFC Software System hierarchy in descending order are: System, complex, subsystem, component, module, and segment.

TERM	DESCRIPTION
System	The CFC Software System is the software system developed under Contract DOT-FA77WA-3955. It consists of two complexes: the Operational Software Complex and the Support Software Complex. These complexes are specified in Volumes I through V of Report Number FAA-RD-76-157, Central Flow Control Computer Program Specifications, in Report Number FAA-RD-76-144, A Computer Program Functional Design of the Simulation Subsystem of an Automated Central Flow Control System, and in Computer Sciences Corporation Report No. CSC/SD-77/6093, Central Flow Control Program Design Specifications. The phrase "the System" is used in place of the lengthier phrase "the Central Flow Control Software System."
Complex	When executing together to process request messages, the Executive Subsystem, the Data Base Management Subsystem, the Application Subsystem, and the Simulation Subsystem are collectively called the Operational

TERM	DESCRIPTION
	Software Complex. The Support Software Complex consists of four subsystems: the System Development Subsystem, the System Generation Subsystem, the System Testing Subsystem, and the System Auditing Subsystem
Subsystem	A subsystem is a set of components that collectively performs one of the functions specified under "Complex" as defined in the documents listed under "System"
Component	A component is a set of modules or modules and components that performs a function. The modules in a component may or may not belong to the same subsystem. Examples of components are (1) a set of modules that is used to request and retrieve data from the data base and (2) the set of modules needed to process the arrival demand (DEMA) request message
Module	A module is a set of code with all of the following characteristics: • It is a program unit that is discrete and identifiable with respect to a compiler and/or an assembler • It has a unique name • It is invocable • It consists of segments (q.v.) • It consists of no more than 200 source statements • It has one entry point and one exit point
Segment	A module consists of one or more segments. A segment is defined as follows: • It is a sequence of contiguous executable statements in which all statements in the segment will be executed if and only if the first statement is executed • It begins with a statement to which control

TERM	DESCRIPTION
Segment (cont'd)	can be transferred and ends with a statement which transfers control to an adjacent segment. An entry segment has no predecessor segments in the module, and an exit segment results in termination or return of control to a calling program

The following terms are used to describe the combination of modules, the basic building blocks of the System, into executable entities.

TERM	DESCRIPTION
load module	A load module is a set of one or more modules combined into a single piece of executable code, ready for loading and execution. Note that a load module is not a particular type of module
transaction load module	A transaction load module is a load module (q.v.) created to perform all the input, retrieval, transformation, update, and output functions to process completely one of the messages specified in the CPFS
build	A build is a component with the following characteristics: • The modules in a build, when loaded and tested, satisfy some subset of the set of requirements for the System • Builds are usually discussed in terms of sequences of builds. In this case, a build contains all the modules of the previous build plus new modules. A build in a sequence satisfies all the requirements satisfied by the previous build plus new requirements
PE	Program element - an instance of execution of a scheduled load module

1.4.2 CODASYL Terms Used to Define Data Hierarchy

The following CODASYL terms, listed in descending order, define the data hierarchy used in System documentation: set, record, data-aggregate, vector, repeating group, and data-item.

TERM	DESCRIPTION
set	A named collection of record types; as such, it establishes the characteristics of an arbitrary number of occurrences of each record type specified
record	A named collection of 0, 1, or more data-items or data-aggregates; there may be an arbitrary number of occurrences of each record type specified
data-aggregate	A named collection of data-items within a record. There are two types of data-aggregates: vector and repeating group
vector	A one-dimensional ordered collection of data-items, all of which have identical characteristics
repeating group	A collection of data that occurs an arbitrary number of times within a record occurrence; the collection may consist of data items, vectors, and repeating groups
data-item	The smallest unit of named data. An occurrence of a data-item is a representation of a value

1.4.3 Glossary of CFC Terms

The first column of the glossary contains CFC terms in alphabetical order. The second column contains a code indicating the category to which the term belongs. The third column contains a description of the term. The correspondence of codes to categories is defined as follows:

Code	Category
S	Subsystem
C	Component
MOD	Module
D	Data
MES	Message
PH	Common Phrase or Term
MIS	Miscellaneous

TERM	CATEGORY CODE	DESCRIPTION
ACID	D	Aircraft ID
ACTV	MES	Activate Inhibited Processing Message
AIRS	MIS	Airport Information Retrieval System
AP	S	Application Subsystem
ARF	MIS	Airport Reservation Function
ARRD	MES	Arrival Delay Prediction Message
ARTCC	MIS	Air Route Traffic Control Center
ASCII	MIS	American Standard Code for Information Interchange
ATCSCC	MIS	Air Traffic Control System Command Center
BA	C	BAL Assembler Component
BAL	MIS	Basic Assembly Language
build	PH	A build is a component with the following characteristics: a. The modules in a build, when loaded and tested, satisfy some subset of the set of requirements for the System. b. Builds are usually discussed in terms of sequences of builds. In this case, a build contains all the modules of the previous build plus new modules. A build in a sequence satisfies all the requirements satisfied by the previous build plus new requirements
CAPL	MES	List Landing Capacities Message
CAPS	MES	Set Landing Capacities Message
CARF	MIS	Central Altitude Reservation Function
CCP	MIS	Contingency Command Post
CE	MIS	Computing Element (q.v.)
CFC	MIS	Central Flow Control

TERM	CATEGORY CODE	DESCRIPTION
CFCF	MIS	Central Flow Control Function
complex	PH	When executing together to process request messages the Executive Subsystem, the Data Base Management Subsystem, the Application Subsystem, and the Simulation Subsystem are collectively called the Operational Software Complex. The Support Software Complex consists of four subsystems: The System Development Subsystem, the System Generation Subsystem, the System Testing Subsystem, and the System Auditing Subsystem
component	PH	A component is a set of modules or modules and components that performs a function. The modules in a component may or may not belong to the same subsystem. Examples of components are (1) a set of modules that is used to request and retrieve data from the data base, and (2) the set of modules needed to process the arrival demand (DEMA) request message
Computing Element	MIS	CPU in the IBM 9020 hardware configuration
CPFS	MIS	Computer Program Functional Specification
CR	C	Conversational Remote Job Entry (CRJE) Component
CRJE	PH	Conversational Remote Job Entry
CSC	MIS	Computer Sciences Corporation
CXSD	MES	Delete Flight Plan Message
data-aggregate	PH	A named collection of data-items within a record. There are two types of data-aggregate: vector and repeating group
data-item	PH	The smallest unit of named data. An occurrence of a data-item is a representation of a value
DA	C	System Data Assembler Component

TERM	CATEGORY CODE	DESCRIPTION
DB	S	Data Base Management Subsystem
DD	MIS	Data Definition
DEMA	MES	Arrival Demand (Today) Message
DEMD	MES	Departure Demand (Today) Message
DESA	MES	Arrival Demand (Future) Message
DESD	MES	Departure Demand (Future) Message
DLDY	MES	Departure Delay Test Message
DM	MES	Departure Message
DTE	MIS	Data Terminal Equipment
EA	C	Environment Adaptation Component
EX	S	Executive Subsystem
FAA	MIS	Federal Aviation Administration
FAD	MIS	Fuel Advisory Departure
FADF	MES	Fuel Advisory Departure Individual Estimated Departure Clearance Times Message
FADP	MES	Fuel Advisory Departure Block Time Message
FADT	MES	Fuel Advisory Departure Test Message
FIXL	MES	Fix Loading Message
FP	MES	Flight Plan Message
FPSD	MES	Add Flight Plan Message
GAEL	MES	List General Aviation Estimates Message
GAES	MES	Set General Aviation Estimates Message
HA	C	Houston Automatic Spooling Program (HASP) Component
HASP	MIS	Houston Automatic Spooling Program

TERM	CATEGORY CODE	DESCRIPTION
INHB	MES	Inhibit Processing Message
Input/Output Control Element	MIS	CPU in the IBM 9020 hardware configuration having a limited instruction set and which can otherwise function as a CE
JAVS	MIS	JOVIAL Automated Verification System
JAX	MIS	Jacksonville, Florida
JC	C	JOVIAL Compiler Component
JCL	MIS	Job Control Language
JI	C	JOVIAL Source Include Processor Component
JL	C	JOVIAL Library Capability Component
JOVIAL	MIS	Jules' Own Version of an International Algorithmic Language - Implementation Language for the CFC System
JP	C	JOVIAL Processors Component
JS	C	JOVIAL Structured Programming Listing Formatter Component
LE	C	Library Edit Component
LIFP	MES	List Flight Plan Message
LISA	MES	List Arrival Message
LISD	MES	List Departure Message
LM	C	Library Maintenance Component
load module	PH	A load module is a set of one or more modules combined into a single piece of executable code, ready for loading and execution. Note that a load module is <u>not</u> a particular type of module
MA	C	Automated Code Auditor Component
MC	C	Maintenance and Certification Component

TERM	CATEGORY CODE	DESCRIPTION
module	PH	A module is a set of code with all of the following characteristics: 1. It is a program unit that is discrete and identifiable 2. It has a unique name 3. It is invocable 4. It consists of segments (q.v.) 5. It consists of no more than 200 source statements 6. It has one entry point and one exit point
MVT	MIS	Multiprogramming with a Variable Number of Tasks
NAFEC	MIS	National Aviation Facilities Experimental Center
NAS	MIS	National Airspace System
NOSS	MIS	NAS Operational Support System (Operating System)
OAG	MIS	Official Airline Guide
OPCX	MIS	Operational Software Complex
OS	MIS	Operating System
OS	C	OS/9020 Component
PDL	MIS	Program Design Language
PDS	MIS	Program Design Specification
PE	MIS	Program Element (q.v.)
Program Element	MIS	An instance of execution of a scheduled load module
PSL	MIS	Programming Support Library
QA	MIS	Quality Assurance
QFLW	MES	Quota Flow First Tier Message
QFLZ	MES	Quota Flow by Zone Message

TERM	CATEGORY CODE	DESCRIPTION
RA	C	Data Reduction and Analysis Component
record	PH	A named collection of 0,1, or more data-items or data-aggregates; there may be an arbitrary number of occurrences of each record type specified
reentrant	MIS	A PE that can have concurrent instances of execution
repeating group	PH	A collection of data that occurs an arbitrary number of times within a record occurrence; the collection may consist of data-items, vectors, and repeating groups
RS	MES	Remove Strip Message
SA	S	System Auditing Subsystem
SB	C	System Build Component
SD	S	System Development Subsystem
SDD	MIS	Software Design Document
SE	MIS	Storage Element in 9020 hardware configuration
segment	PH	A module consists of one or more segments. A segment is defined as follows: 1. It is a sequence of contiguous executable statements for which all statements in the segment will be executed if and only if the first statement is executed 2. It begins with a statement to which control can be transferred and ends with a statement which transfers control to an adjacent segment. An entry segment has no predecessor segments in the module, and an exit segment results in termination or return of control to a calling program.
set	PH	A named collection of record types; as such, it establishes the characteristics of an arbitrary number of occurrences of each record type specified

TERM	CATEGORY CODE	DESCRIPTION
SG	S	System Generation Subsystem
ST	S	System Testing Subsystem
serially reusable	MIS	A PE that cannot have concurrent instances of execution but which can be executed serially without refreshing (reloading)
subsystem	PH	A subsystem is a set of components that collectively performs one of the functions specified under "Complex" as defined in the documents listed under "System"
System	PH	The CFC Software System is the software system developed under Contract DOT-FA-77WA-3955. The CFC Software System consists of two complexes: the Operational Software Complex and the Support Software Complex. These complexes are specified in Volumes I through V of Report Number FAA-RD-75-157, Central Flow Control Computer Program Specifications and in Report Number FAA-RD-76-144, A Computer Program Functional Design of the Simulation Subsystem of an Automated Central Flow Control System. The phrase "the System" shall be written in place of the lengthier phrase "the Central Flow Control Software System"
TA	C	Online Test Director Component
TLM	MIS	Transaction Load Module
TM	C	THREADS Component
TO	MIS	Technical Officer
TR	C	Test Case Generator Component
transaction load module	PH	A transaction load module is a load module (q.v.) created to perform all the input, retrieval, transformation, update, and output functions to process completely one of the messages specified in the CPFS

TERM	CATEGORY CODE	DESCRIPTION
TS	C	Time Sharing Option (TSO) Component
TSO	PH	Time Sharing Option
UT	C	Utilities Component
vector	PH	A one-dimensional ordered collection of data-items, all of which have identical characteristics

SECTION 2 - SYSTEM ARCHITECTURE

Section 2 is divided into two subsections: 2.1, System Description, which contains an overview of the System and which is intended to give the reader a brief explanation of what the System does; and 2.2, Subsystem Descriptions, which describes the software architecture and data flow for the four subsystems of the Operational Software Complex (OPCX) and the four subsystems of the Support Software Complex (SPCX) using hierarchy diagrams.

2.1 SYSTEM DESCRIPTION

The basic CFC Software System is designed to improve the automation of the ATCSCC central flow functions of monitoring and regulating air traffic flow on a national basis by providing capabilities for predicting, detecting, and resolving air traffic problems. The System has a direct digital interface to the ATCSCC for rapid exchange of CFC information. In addition, the System has a digital data channel interface to the 20 National Airspace System (NAS) En Route computers at the ARTCCs.

The 20 ARTCCs generate messages (or transactions) for selected flight plans for forwarding to the CFC computer. Five of the 20 En Route computers are designated as store-and-forward focal points to concentrate data from several centers before transmission to the CFC computer. Two En Route computers are designated as relays to pass data on to the nearest store-and-forward focal point. Thus, CFC messages between En Route facilities utilize existing computer communication links. New links will be established between the En Route computers at the five store-and-forward ARTCCs and the CFC computer.

Figure 2.1-1 shows this CFC computer interface configuration and identifies the locations of the store-and-forward computers and the two relay computers.

2.1.1 Message Processing

The basic CFC System processes 26 application messages. These can be grouped into four major categories: count, list, simulation, and data base update. The following paragraphs identify and briefly describe the 26 messages.

2.1.1.1 Count Messages

The count message category consists of six messages that cause the CFC data base to be searched and specific counts of data to be tabulated. The messages are:

- DEMD (departure demand), which tabulates hourly counts of aircraft departing a designated airport(s) on the current day.
- DEMA (arrival demand), which tabulates hourly counts of aircraft scheduled to arrive at a designated airport(s) on the current day. Subtotals for currently airborne aircraft are included.
- DESD (future departure demand), which tabulates hourly counts of aircraft departing a designated airport(s) on a future day(s).
- DESA (future arrival demand), which tabulates hourly counts of aircraft that are scheduled to arrive at a designated airport(s) on a future day(s).
- FIXL (fix loading), which tabulates hourly counts of aircraft scheduled to cross a designated (adapted) fix(es). Subtotals for currently airborne aircraft are included.

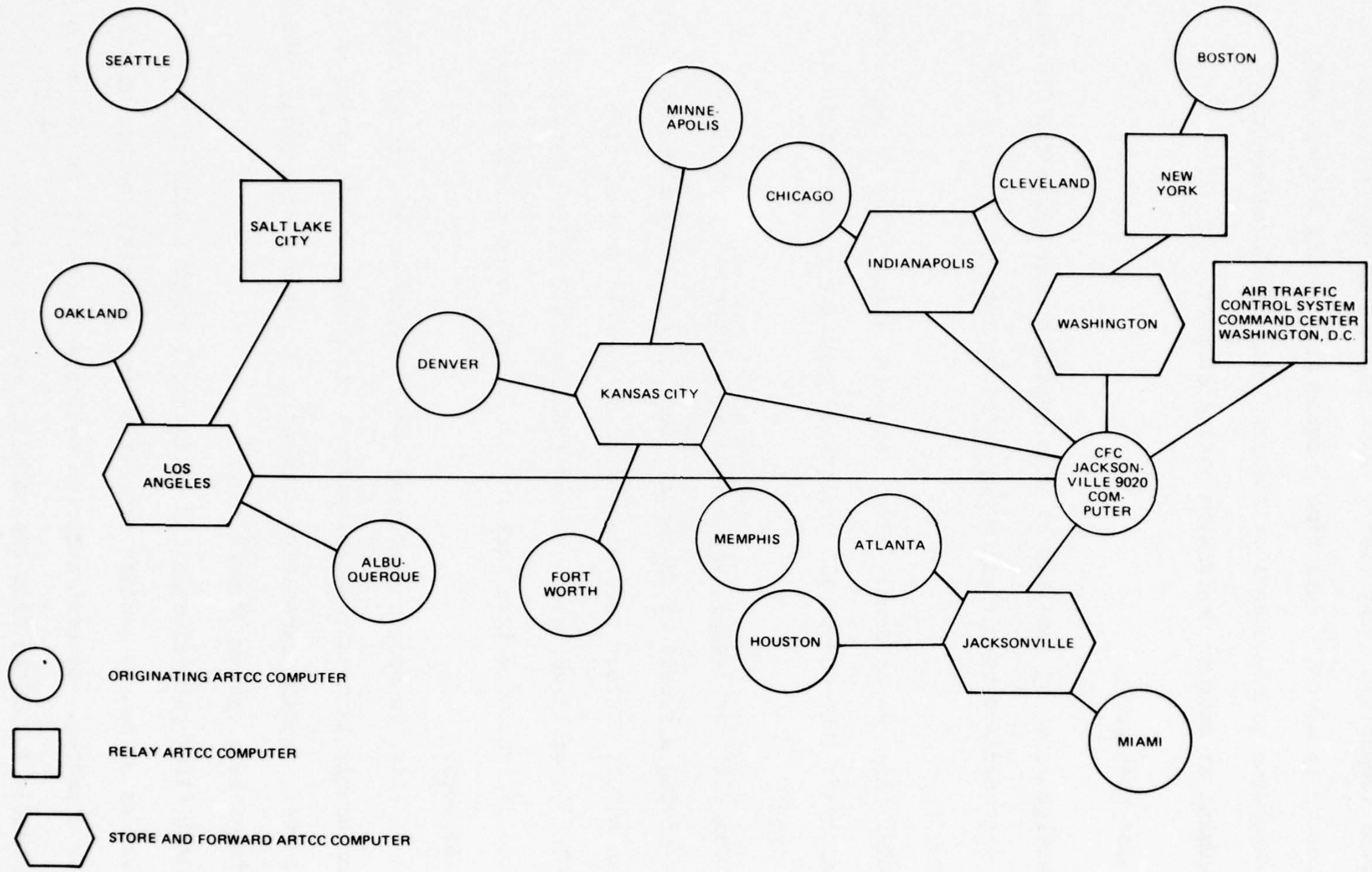


Figure 2.1-1. Central Flow Control System Computer Network

- DLDY (departure delay test), which applies an entered delay factor to aircraft departing a designated pacing airport and tabulates hourly counts of the resulting predicted arrival demand at another designated pacing airport.

2.1.1.2 List Messages

The list message category consists of five messages that cause the CFC data base to be searched and specifically requested data to be listed. The messages are:

- LISD (list departures), which tabulates information on departing aircraft from one or all designated pacing airports within an ARTCC.
- LISA (list arrivals), which tabulates information on planned arriving aircraft at one or all designated pacing airports within an ARTCC. Output data are sorted by planned arrival time.
- LIFP (list flight plan), which tabulates all active data for the designated flight that involve a designated or all pacing airports.
- CAPL (list landing capacities), which tabulates the stored values of hourly landing capacities for a designated or all pacing airports. Landing capacities currently in use are separately tabulated from the normally adapted values.
- GAEL (list general aviation estimates), which tabulates the stored values of hourly landing capacities for a designated pacing airport or a center. General aviation estimates currently in use are separately tabulated from the normally adapted values.

2.1.1.3 Simulation Messages

The simulation message category consists of six messages that cause delay and/or capacity predictions to be computed and tabulated. The messages are:

- ARRD (arrival delay prediction), which computes anticipated arrival delays for flights arriving at a specified airport. Output data are ordered by hourly prediction, current airborne flights are identified and average as well as anticipated peak delay data are included.
- FADF (fuel advisory departure (FAD) for estimated departure clearance times (EDCT)), which generates the information obtained via the FADP message but, in addition, generates output data by flight to be used for "energy conservation" flow control.
- FADP (fuel advisory departure block time), which is used in conjunction with FAD procedures. Output data include the landing capacities used in the computations, "quota" reports based on stipulated holding stack sizes, and EDCT block time assignments.
- FADT (fuel advisory departure test), which generates a one-time output, typically used to test and evaluate the need for implementing FAD procedures.
- QFLW (quota flow first tier), which assigns first tier quotas with respect to an identified pacing airport based on known traffic demand and landing capacity.

- QFLZ (quota flow by zone), which assigns quotas with respect to an identified pacing airport based on the requested zone and known traffic demand and landing capacity.

2.1.1.4 Data Base Update Messages

The data base update message category consists of nine messages that change or augment the data base. One group of these messages are entered by flow control personnel; the second group are automatically generated by the NAS En Route Stage A System. The group of messages entered by flow control personnel are:

- ACTV (activate flight plan), which cancels the inhibition of data retrieval on a specified airline or individual flight effective the current date or a specified date.
- INHB (inhibit flight plan), which inhibits the retrieval of data for a specified airline or individual flight effective the current date or a specified date.
- FPSD (add flight plan), which permits ATCSCC personnel to add air carrier flight plan(s) to the data base.
- CXSD (delete flight plan), which cancels identified flight data from the data base.
- CAPS (set landing capacities), which replace the current values of landing capacities for the identified pacing airport for the stipulated time interval.
- GAES (set general aviation estimates), which replaces the current values of landing capacities for the identified pacing airport or center for the stipulated time interval.

The data base update messages generated by the NAS En Route Stage A System are:

- FP (flight plan), which forwards non-air-carrier flight plan data whenever a flight plan is entered into NAS for a flight that is scheduled to land at a pacing airport.
- DM (departure), which forwards the actual departure time for any flight that is planning to land at a pacing airport.
- RS (remove strip), which provides notification that a proposed departure flight that is scheduled to land at a pacing airport is cancelled.

2.1.2 System Overview

The CFC System provides real-time response to all messages requested by ATCSCC personnel and an offline capability for software development, maintenance, and data reduction and analysis. To satisfy these functional requirements, the System has been organized into two separate complexes: the Operational Software Complex (OPCX) and the Support Software Complex (SPCX). The following subsections describe the OPCX and SPCX capabilities, data flow, and organization.

The medium for interfacing the ATCSCC in Washington, D. C., and the 20 ARTCCs located nationwide with the CFC System via communications is the set of 26 messages (or transactions). Figure 2.1-2 presents the CFC System overview and identifies the sources, destinations, and types of information being passed through the System. Major inputs include the Official Airline Guide

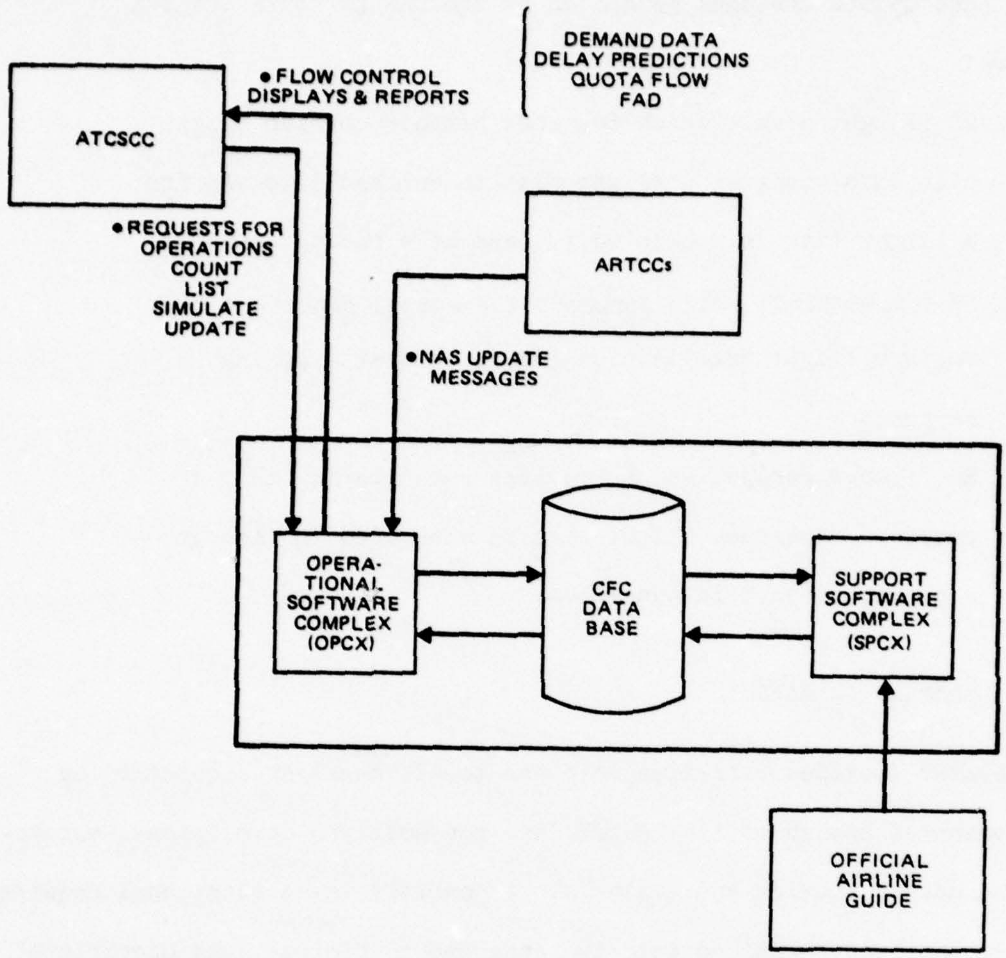


Figure 2.1-2. Central Flow Control System Overview

(OAG) tape and the transaction messages. The OAG tape contains information (aircraft identification, arrival and departure terminals, and scheduled departure time) on scheduled aircraft flights. This OAG information is used by the Support Software Complex to produce the CFC data base. The ATCSCC sends requests for operations messages (count, list, simulate, update) and the ARTCCs send En Route update messages to the System's Operational Software Complex via communications links. The ATCSCC may send any of the previously identified 23 operations messages, whereas the ARTCCs may send only the flight plan, departure, and remove strip messages. These last three En Route update messages will augment the OAG information in the data base by providing data relating to non-scheduled flights and current departure times or cancellations.

The CFC Software System will provide the Command Center with air traffic reports and displays consisting of demand data, delay predictions, quota flow, and FAD.

2.1.2.1 System Hardware

Two separate hardware configurations for support of the overall CFC System have been specified. The first is a dual processor IBM 9020 configuration, which will be used in support of the automated operational CFC functions. It is referred to as the operational duplex. The second configuration is a single processor 9020 and is referred to as the redundant simplex. The redundant simplex is primarily to process offline functions of the System, including data base assembly, system development, maintenance, performance analysis, and testing. Figure 2.1-3 illustrates the major elements of the two configurations.

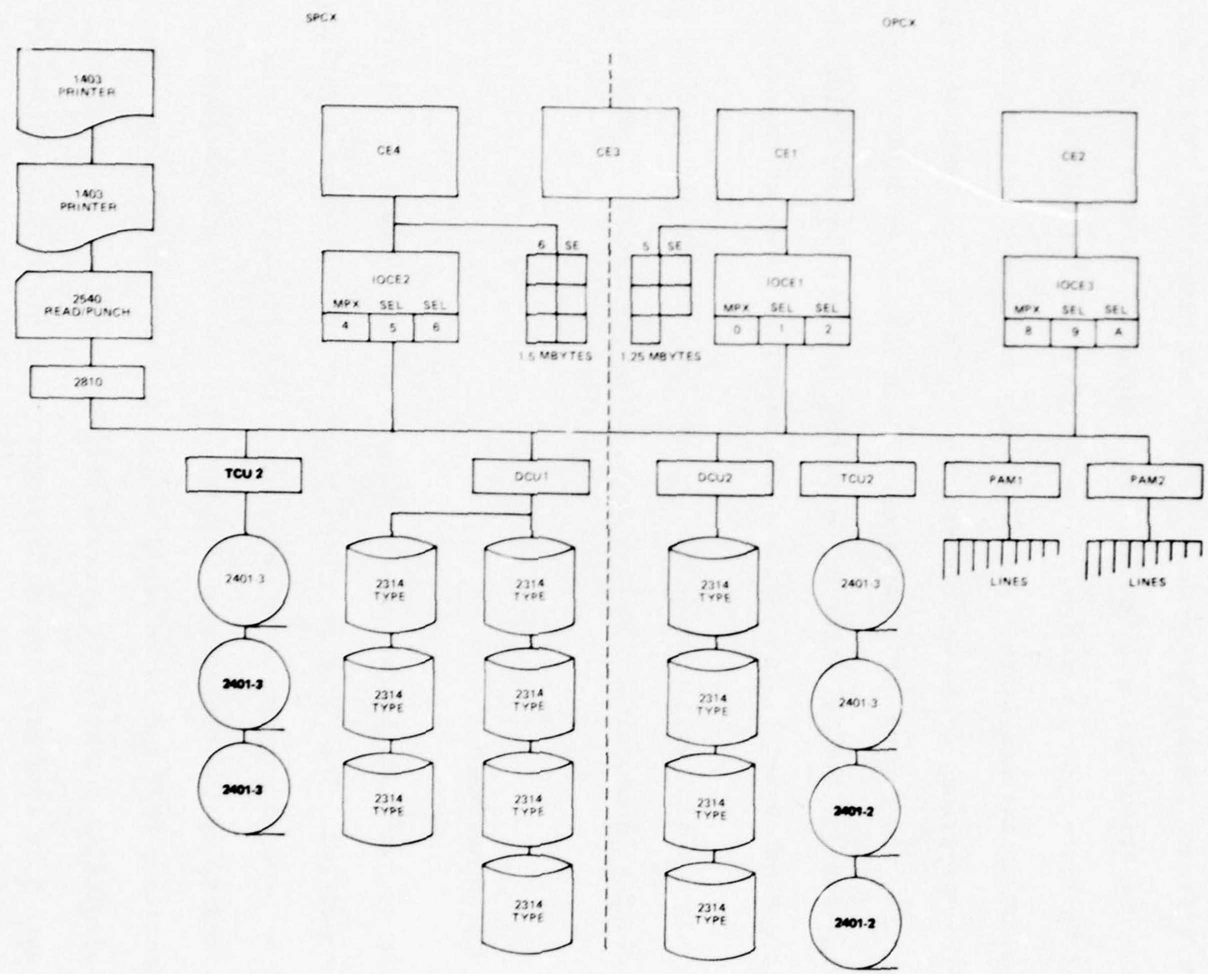


Figure 2.1-3. Central Flow Control System Hardware

The duplex has:

- Two 9020 compute elements (CEs)
- Two 9020 I/O compute elements (IOCEs)
- Five storage elements (SEs)
- One disk control unit (DCU)
- Four 2314 disk drives
- One tape control unit (TCU)
- Four tape drives
- Two peripheral adapter modules (PAMs)

The simplex has:

- One 9020 compute element
- One 9020 I/O compute element
- Five storage elements
- One disk control unit
- Seven 2314 disk drives
- One tape control unit
- Three tape drives

For operational flexibility, one compute element, one card reader, and two printers can be shared between the duplex and the simplex. The reader and the printer are dynamically switchable through the operator's console to any IOCE.

2.1.2.2 System Functions

The CFC System has been organized functionally into two separate complexes: OPCX and SPCX. The OPCX functionally interfaces with the Central Flow Controller in the ATCSCC and directly responds to requests to dynamically adjust the System data base; to generate demand, delay, QFLOW, and FAD reports; and to disseminate flow control advisories and information messages. The principal functions of the OPCX are (1) the receipt of input messages, (2) the control of message processing, (3) the performance of message-specific application processes, and (4) the output of message processing results. These four principal functions must be applied for all message types. The processing within each major function, which is described in later sections, varies significantly depending on the source/destination and its intended operational purpose.

The SPCX provides the capabilities required in support of the development/ maintenance, generation, testing, and auditing of the CFC software. The SPCX performs the following system-level functions: (1) process input source code for development and checkout of programs via utility processes; (2) create, modify, and integrate new elements (parameter structures and/or software); (3) detect and diagnose problems and summarily analyze and report on the performance of the OPCX (peripheral hardware and software); (4) test, through the use of near-real-world simulation processes, the operation of all functions of the OPCX; and (5) monitor and report on the development project progress through effective management aid processes.

2.1.2.3 System Structure

The System structure and interdependencies of the two complexes are illustrated in Figure 2.1-4.

To satisfy the functions of transaction message processing, the OPCX is organized into four major subsystems: Executive Subsystem, Data Base Management Subsystem, Application Subsystem, and Simulation Subsystem. The Simulation Subsystem can be thought of as part of the Application Subsystem in the sense that the simulation processing is in response to messages from the ATCSCC and produces reports output to the ATCSCC.

The capabilities required in support of the development, generation, testing, and auditing functions are many and varied. The SPCX is therefore represented by a collection of functions that are not integrally related, but stand alone to provide individual capabilities (e.g., testing and source code compilation). In most cases, each function provides a useful capability independent of the existence of the others, but in no instance could a major function be eliminated without seriously degrading the quality of the overall support capabilities. To satisfy these functions, the SPCX is organized into four major subsystems: System Development Subsystem, System Generation Subsystem, System Testing Subsystem, and System Auditing Subsystem.

2.1.2.3.1 OPCX Structure

The Executive Subsystem provides hardware resource management, communications handling, transaction message dispatching, I/O request servicing, and the relation of logical to physical device assignment.

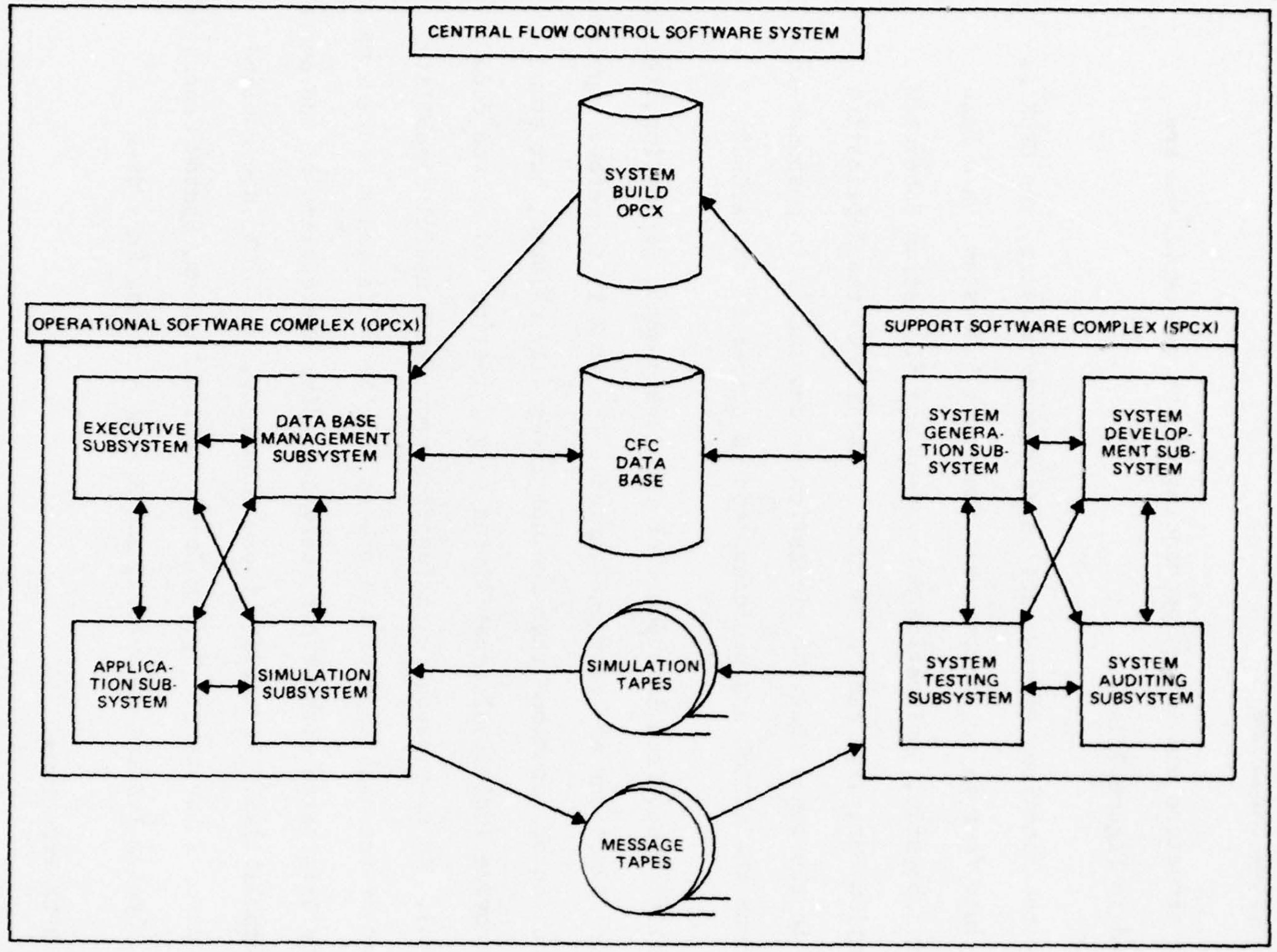


Figure 2.1-4. Central Flow Control System Structure

The Data Base Management Subsystem relates the logical I/O requests from the Application Subsystem and the Simulation Subsystem to specific I/O requests, and presents these requests to the Executive Subsystem for servicing.

The Application Subsystem is responsible for all aspects of message processing (i.e., input message processing, data base retrieval processing, transform processing, data base update processing, and output message processing). The interface with the Executive Subsystem is on a logical-request basis, requiring no knowledge, on the part of the Application Subsystem, as to how a request will be serviced. The interface with the Data Base Management Subsystem is accomplished through storage and retrieval requests.

The Simulation Subsystem processes the simulation messages and interfaces with the Executive and Data Base Subsystems in the same manner as the Application Subsystem.

2.1.2.3.2 SPCX Structure

The System Development Subsystem consists of operating-system-level software, including language processors, needed for development of the System.

The System Generation Subsystem consists of Library Maintenance, Environment Adaptation, System Build, and System Data Assembler software.

The System Testing Subsystem consists of an automated system for verifying software written in JOVIAL, a test case generator, and an online test director.

The System Auditing Subsystem consists of aids in the areas of data reduction and analysis of past System performance and operation, automated auditing of JOVIAL source code to measure conformance to standards, and management information reporting.

2.2 SUBSYSTEM DESCRIPTIONS

2.2.1 Operational Software Complex (OPCX)

2.2.1.1 OPCX Overview

The Operational Software Complex (OPCX) is the online portion of the Central Flow Control Software System. That is, it directly interfaces with Central Flow Controllers in the Air Traffic Control System Command Center (ATCSCC) and with the Air Route Traffic Control Centers (ARTCCs). Messages coming from the ATCSCC are requests from Central Flow Controllers at Data Terminal Equipment (DTE) consoles for lists of data elements to be retrieved from the data base, for tabulated counts of data in the data base, for anticipated delay and/or capacity predictions (simulations), and for updates to the data base. ARTCC messages are flight plans for non-scheduled flights, actual departure times, and deletions (remove strip) of flight records because of cancellations or diverted flights. Each message coming into the CFC System from the ATCSCC is acknowledged by either an accept or reject message. In addition, most messages from the ATCSCC (all except data base update messages) will result in responses to the initiating DTE consoles providing the requested information. In the basic System implementation, acknowledgement messages are not sent to the originating ARTCCs.

The OPCX consists of the Executive (EX), Data Base Management (DB), Application (AP), and Simulation (SI) Subsystems, as depicted in the hierarchy diagram shown in Figure 2.2.1-1. The Executive Subsystem manages the OPCX hardware and software resources, initiates and terminates tasks in response to incoming messages, and provides input/output services to the other subsystems of the OPCX.

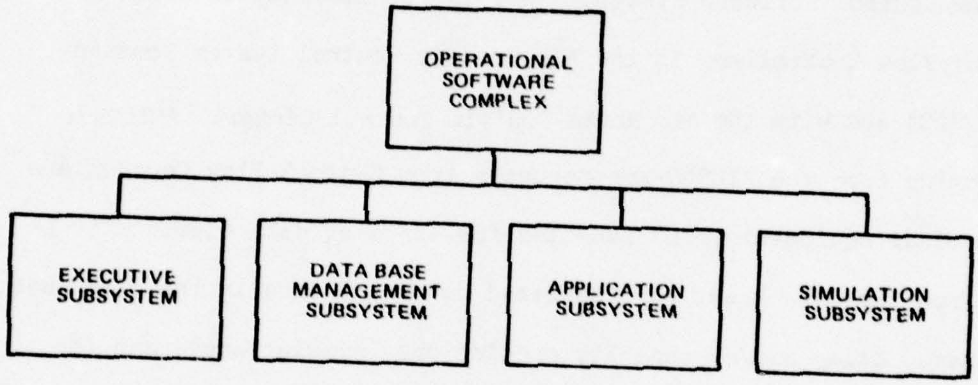


Figure 2.2.1-1. Hierarchy Diagram of the Operational Software Complex

The Data Base Management Subsystem provides access to the CFC data base for the Application and Simulation Subsystems, provides facilities for backup and recovery of the data base in case of failure, and provides the means to ensure data base integrity. The Application Subsystem performs the necessary processing functions to service incoming messages.

The Simulation Subsystem performs simulations in response to requests for predictive information such as Arrival Delay Prediction (ARRD), Quota Flow (QFLW), and Fuel Advisory Departure (FAD).

The basic functions of the OPCX are depicted in the OPCX Overview diagram presented in Figure 2.2.1-2. There are four types of input to the OPCX: the data base, which consists of flight record information and associated data such as airports, centers, fixes, airline operators, and general aviation estimates; NAS ARTCC messages, including flight plans, actual departure times, and remove strip (flight plan deletion) messages; requests from Central Flow Controllers in the ATCSCC to retrieve data, tabulate data, predict future delays or capacities, and update the data base; and control and supervisory messages from the System console. All work performed by the OPCX is in response to messages, either from the ATCSCC, the NAS ARTCCs, or the system console. The functions performed include data base maintenance, data base updating, data base retrieval, tabulation of information extracted from the data base, and simulation of air traffic conditions to develop predictive information.

Figure 2.2.1-3 depicts the complex-level threads for the OPCX. On the first page are the threads for the Executive Subsystem (EX). The first two threads

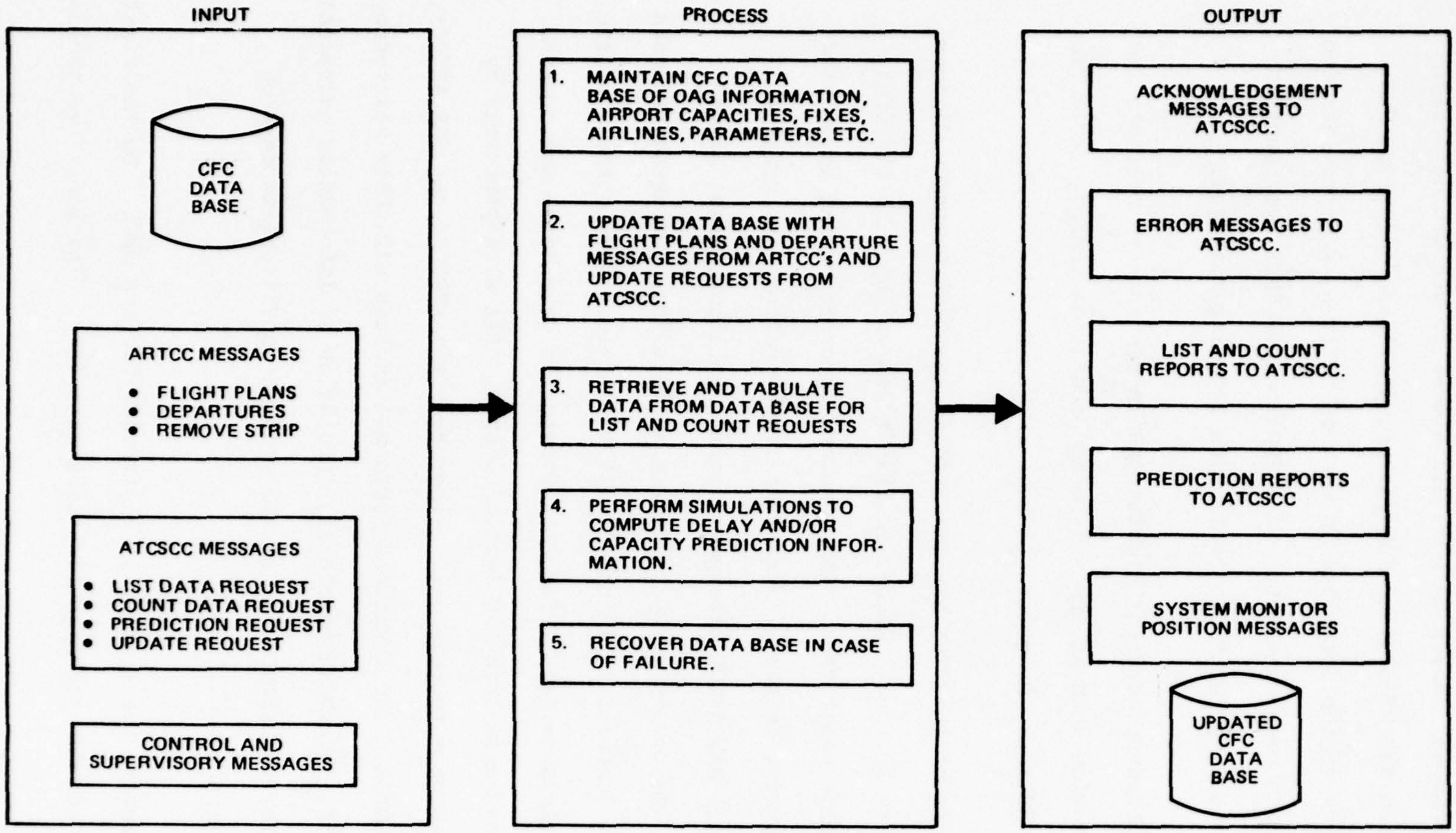


Figure 2.2.1-2. Operational Software Complex Overview

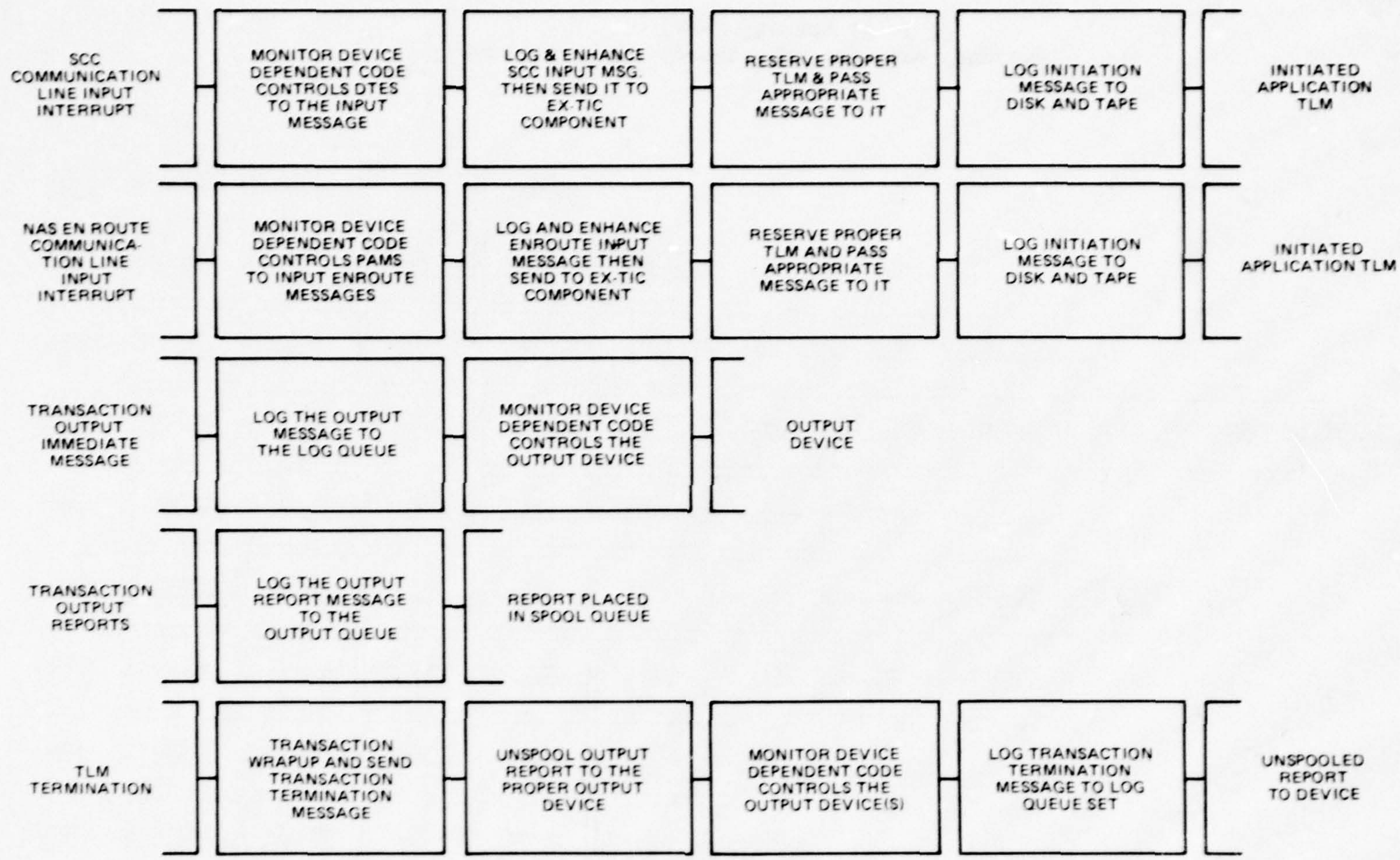


Figure 2.2.1-3. Operational Software Complex-Level Threads (1 of 3)

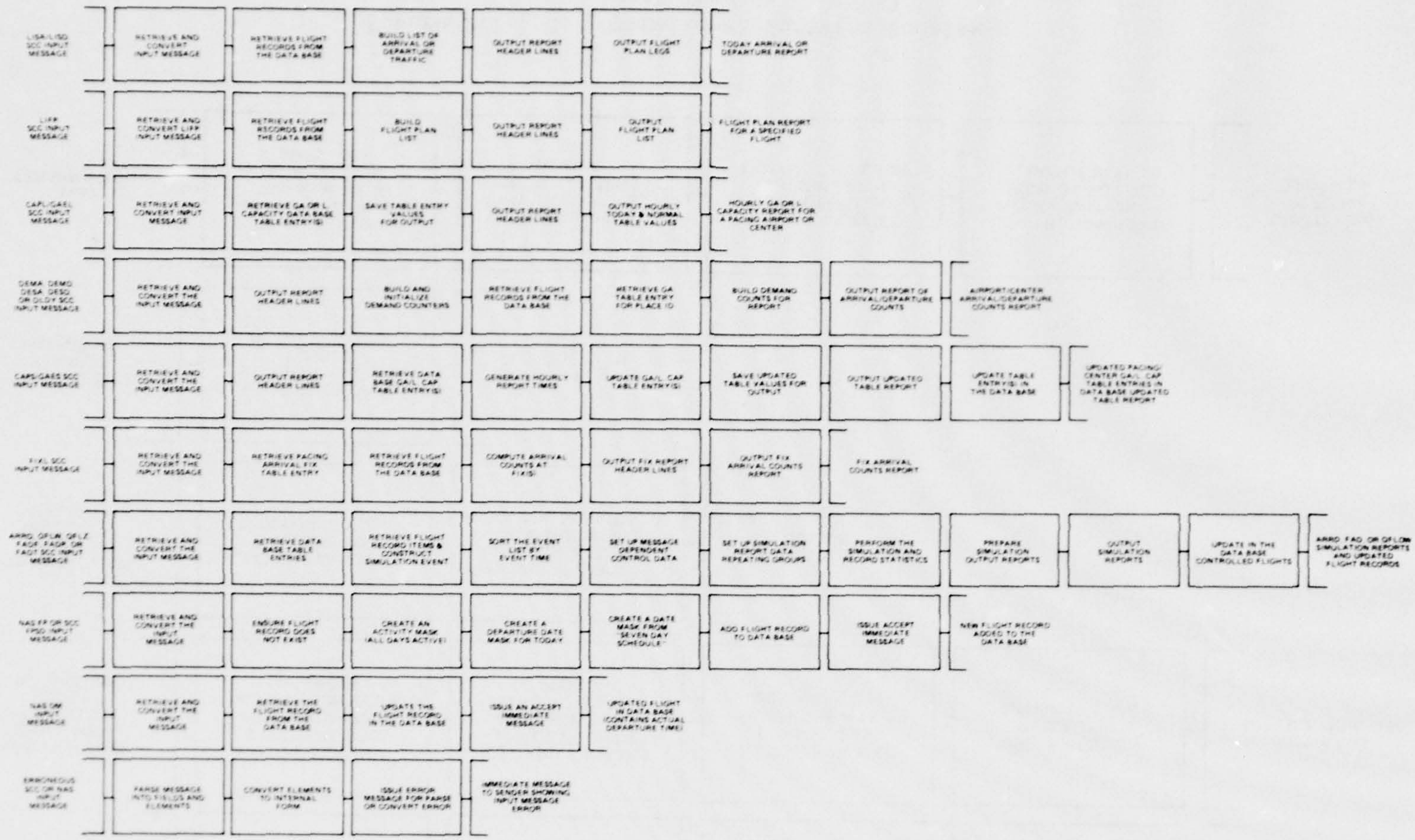


Figure 2.2.1-3. Operational Software Complex-Level Threads (2 of 3)

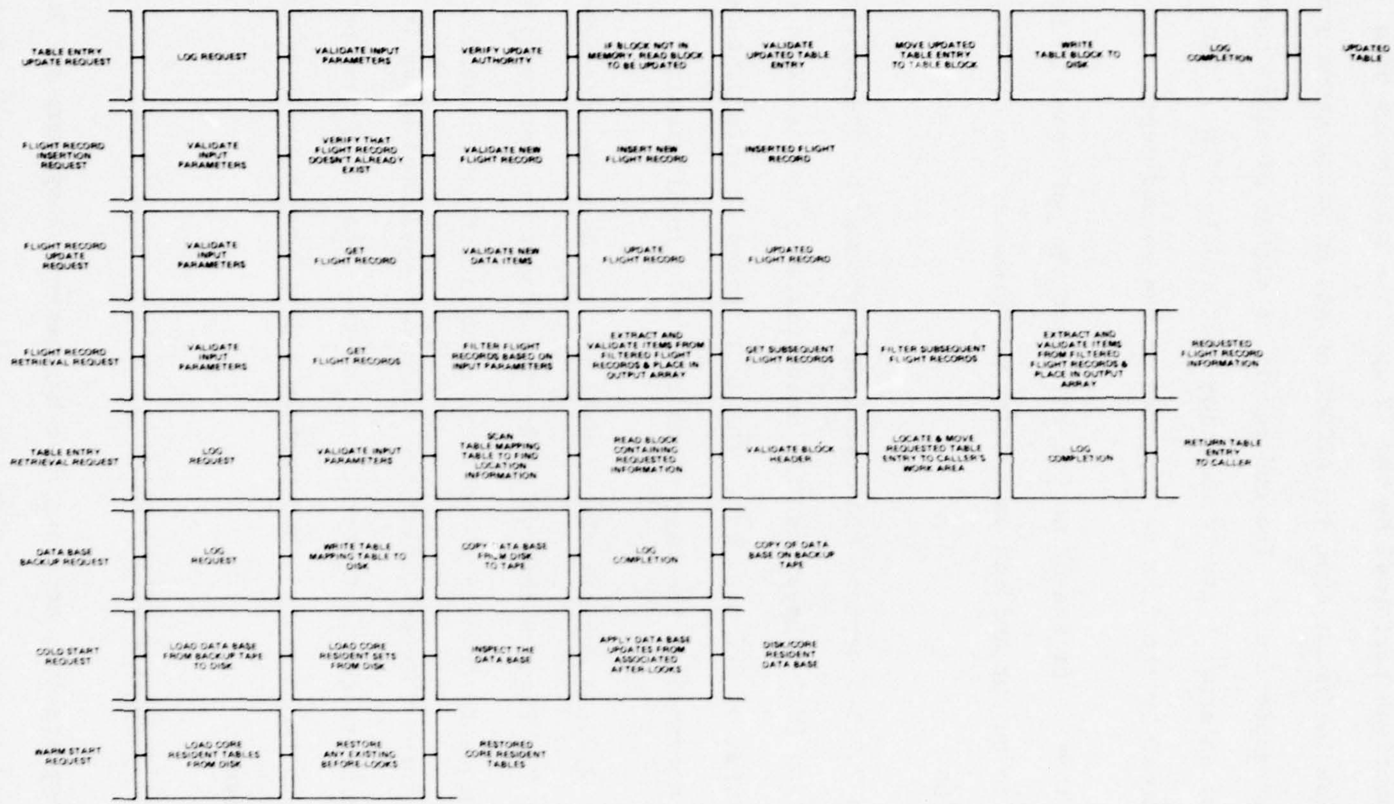


Figure 2.2.1-3. Operational Software Complex-Level Threads (3 of 3)

represent the functions performed by the EX upon the occurrence of a communication line interrupt from the ATCSCC or NAS En Route store and forward centers, respectively. The end result of either of these threads is the initiation of an appropriate transaction load module (TLM). The processing performed by the TLMs is depicted on the second page.

The other three threads of the EX depict the functions performed in response to TLM requests to output and the processing performed at the completion of a TLM execution.

Page 2 of Figure 2.2.1-3 depicts the functions performed by the Application (AP) and Simulation (SI) subsystems TLMs when initiated by the EX in response to the various types of CFC messages. The last thread on this page depicts the processing performed in the case of an erroneous input message.

Page 3 of Figure 2.2.1-3 depicts the threads for the Data Base Management (DB) Subsystem. The first 4 threads depict the processing performed by DB in response to requests from AP to retrieve or update flight record or table data. The last three threads depict the functions performed by DB in response to requests from EX to create a backup copy of the data base, perform a data base start up (cold start), and perform a data base start over (warm start), respectively.

2.2.1.1.1 Logical Structure of the OPCX

The Executive, Application, and Data Base Management Subsystems are logically interrelated in such a way that all physical hardware and external considerations are insulated from the Application and Data Base Management

Subsystems by the Executive Subsystem. In addition, all accesses to the CFC data base are made through the Data Base Management Subsystem. This division of functions ensures a higher degree of software and data reliability than would otherwise be possible, because interfaces are well defined and can be comprehensively tested. A further benefit to this approach is that it facilitates maintenance and future enhancement of the System by isolating the pertinent software affected by a change. For example, should future additions to the data base result in changes to its organization, structure, or physical location (e.g., core or disk), the Application Subsystem modules would not be affected (other than perhaps requiring recompilation with a new COMPOOL definition).

Figure 2.2.1-4 depicts the logical structure of the OPCX. The initiation of activity begins with a message received from either the ATCSCC or one of the ARTCCs. The Executive Subsystem performs the communications handling to receive the message, then determines the type of message and initiates a process within the Application Subsystem to perform the functions required by the message. For any given message, one of these functions will include accessing the data base (either retrieval or update). The Application Subsystem calls upon the Data Base Management Subsystem to satisfy these needs. The Data Base Management Subsystem, in turn, calls upon the Executive Subsystem to access the data base, usually via I/O services.

Once the Application Subsystem obtains the requested data, it performs the necessary computations, simulations, and/or data manipulations, updates the data base (if required), formats a response message (if one is required), and transfers to the Executive Subsystem. The Executive Subsystem then sends the response message to the DTE console where the request message

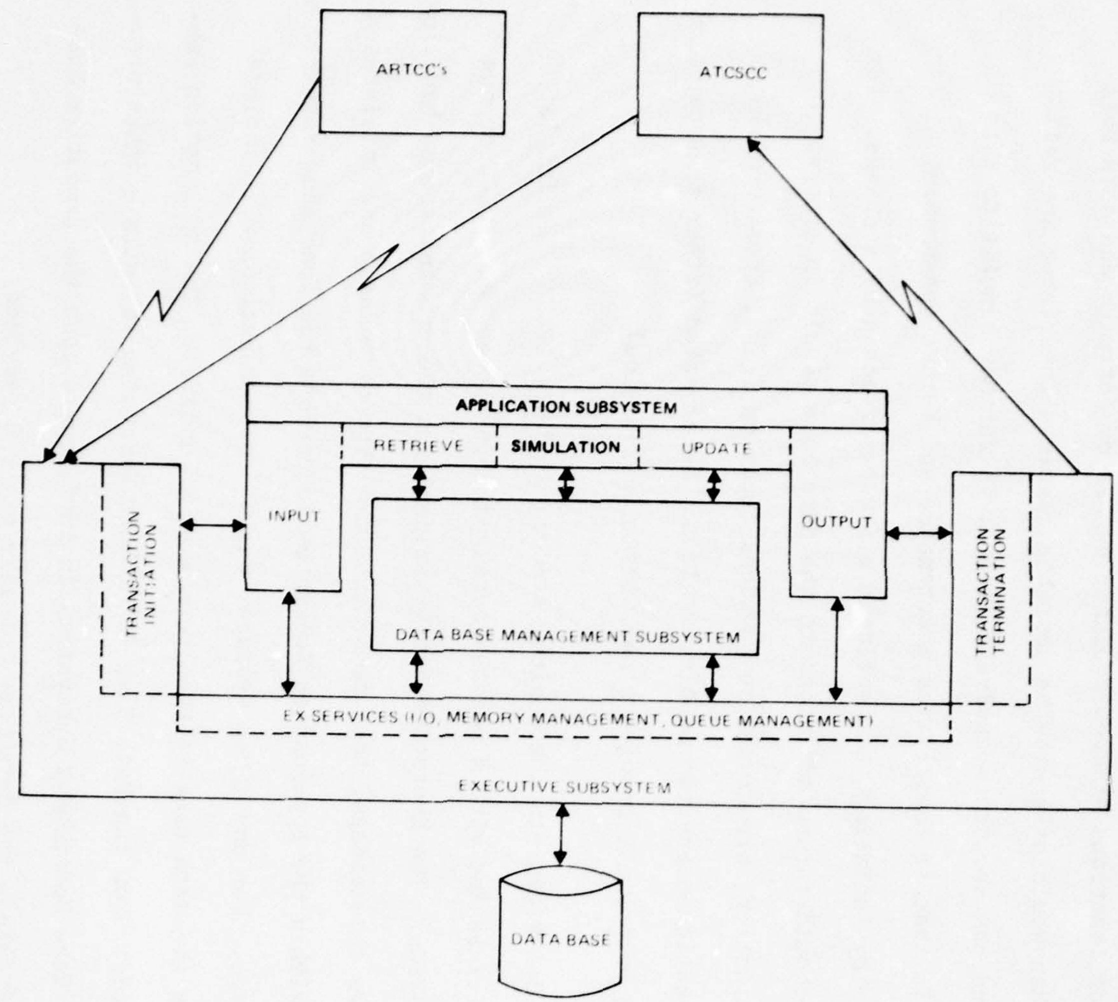


Figure 2.2.1-4. Logical Structure of the Operational Software Complex

originated and/or to an alternate device selected by the Controller.

2.2.1.1.2 Physical Structure of the OPCX

The physical structure of the software within the OPCX is as represented in Figure 2.2.1-5. For clarity, the Application Subsystem is represented as a single collection of software components called Transaction Load Modules (TLMs). At any given time, some of these TLMs are core-resident and some are on disk. They are brought into core as needed by the Executive Subsystem. There are 26 different TLMs in the basic System, each tailored to a specific message type to be processed; for example, there is a LISA TLM and a DEMA TLM. In future versions of the System, there will be additional message types, and additional TLMs to process them. This tailoring is accomplished by linkage-editing only those modules necessary to process a given type of message to obtain a TLM that is specific to its respective message type.

The TLMs include modules from the Data Base Management Subsystem needed to process their respective messages. The Executive Subsystem receives messages from the ATCSCC and ARTCCs, logs them in, classifies them by types, and initiates the appropriate TLMs to process them. The Executive Subsystem also performs I/O services, memory management, etc., as requested by the TLM. The Data Base Management Subsystem modules integrated into each TLM use the I/O services of the Executive Subsystem to access the data base in response to calls from the Application Subsystem modules. Each TLM is structured to contain the Application and/or Simulation Subsystem components necessary to perform the specific processing required by the associated

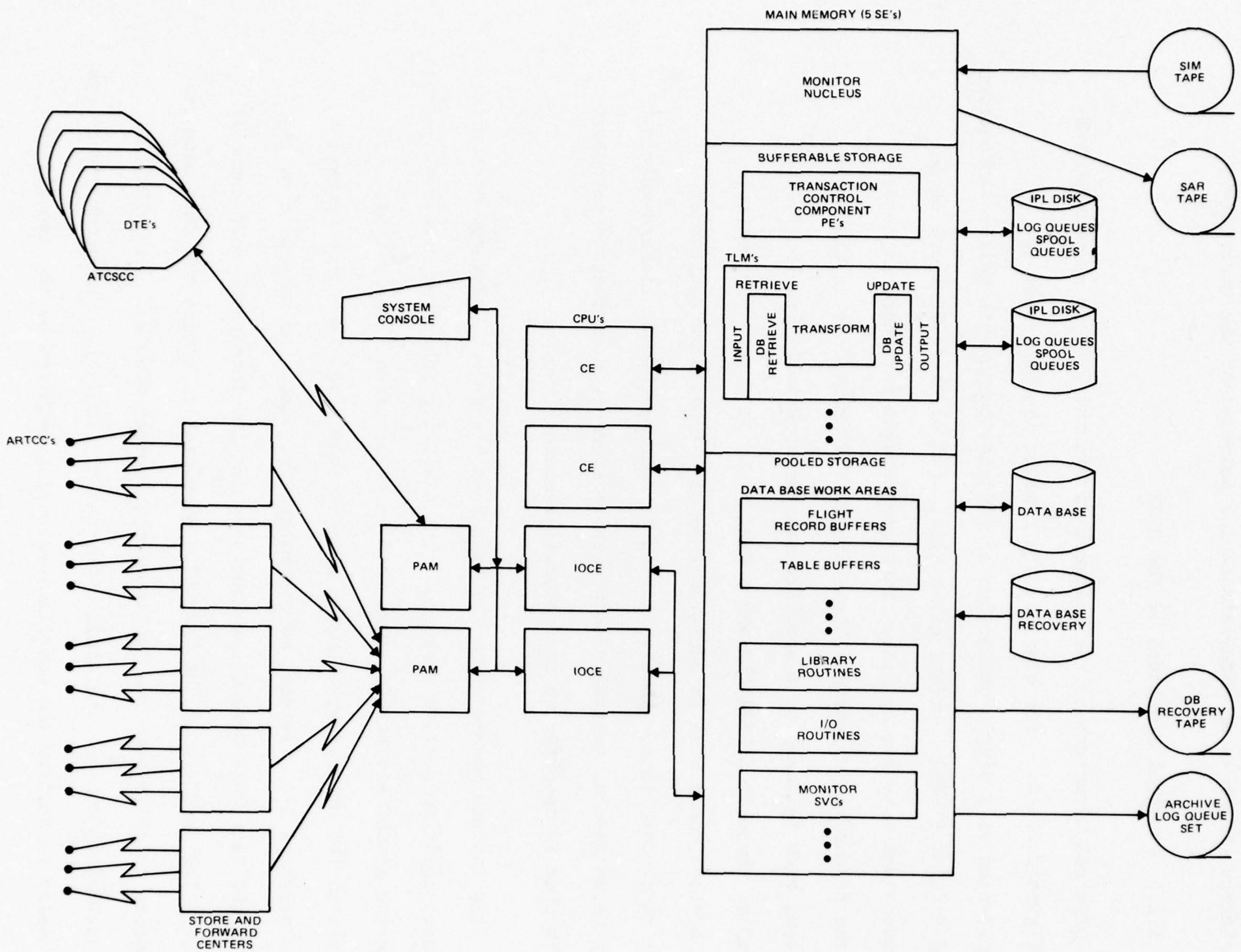


Figure 2.2.1-5. Physical Structure of the Operational Software Complex

message type. These Application, Simulation, and Data Base Management Subsystem components will be invoked as required to perform data base retrieval, data base update, data manipulation, simulation, and requested data output. Control then returns to the Executive Subsystem, which sends the response message to the initiating source (ATCSSC).

2.2.1.2 Executive (EX) Subsystem

a. Purpose. The purpose of the CFC Executive (EX) Subsystem is to provide management of the resources of the OPCX and its associated hardware. The Executive Subsystem dynamically allocates computing, storage, and input/output resources to program elements (PEs) and provides centralized services for transaction control, queuing, and logging as well as OPCX testing.

b. Subsystem Overview. The Executive Subsystem performs as a real-time, transaction-oriented operating system for the IBM 9020A configuration to be used for the execution of the OPCX. The programs to be executed under the control of the Executive Subsystem are transaction-driven and initiated in a multiprocessing machine environment. The processing requirements of the Executive Subsystem are initiated by two general means: the detection by the hardware of some form of interrupt and the detection by the Executive Subsystem of a recognizable type of message. The processing for the first is implemented by the CFC monitor, which resides in main storage, and is activated in the event of an interrupt condition by a hardware facility termed a Load Program Status Word (LPSW). The automatically loaded PSW effects the transfer of processing control to a designated location in main memory where the appropriate instructions reside for the particular interrupt type. There are five types of interrupts: Input/Output, External, Supervisor Call (SVC), Program, and Machine Check. In the event of an Input/Output interrupt, the interrupt processing modules would determine the type of incoming data and cause the initiation of the appropriate transaction processing component. This effects the second means of detecting a processing requirement. The processing of

The interrupts are accomplished by the CFC Monitor Component of the Executive Subsystem, and the subsequent processing of transactions is accomplished by the Transaction Control Component of the Executive Subsystem. These are the two major components of the Executive Subsystem. Figure 2.2.1-6 presents the Executive Subsystem Overview. Figure 2.2.1-7 shows the interaction of the two major components of the Executive Subsystem.

c. Functional Description. The Executive Subsystem performs all the functions necessary to support the management of resources and the processing of transactions into and out of the system. The functions performed by the Executive Subsystem are:

- Supervisor Call (SVC) Processing
- Monitor Startup/Startover
- Program Interrupt Processing
- Date and Time Maintenance
- System Analysis Recording (SAR)
- Test and Debug Processing
- Input/Output Processing
- TCC Startup/Startover
- Transaction Input/Output Processing
- Transaction Initiation/Termination Processing

All processes performed by the Executive Subsystem are included in these functions.

d. Major Components. The Executive Subsystem consists of two major components: the CFC Monitor (MON) Component and the Transaction Control Component (TCC). The MON Component consists of twelve components, which

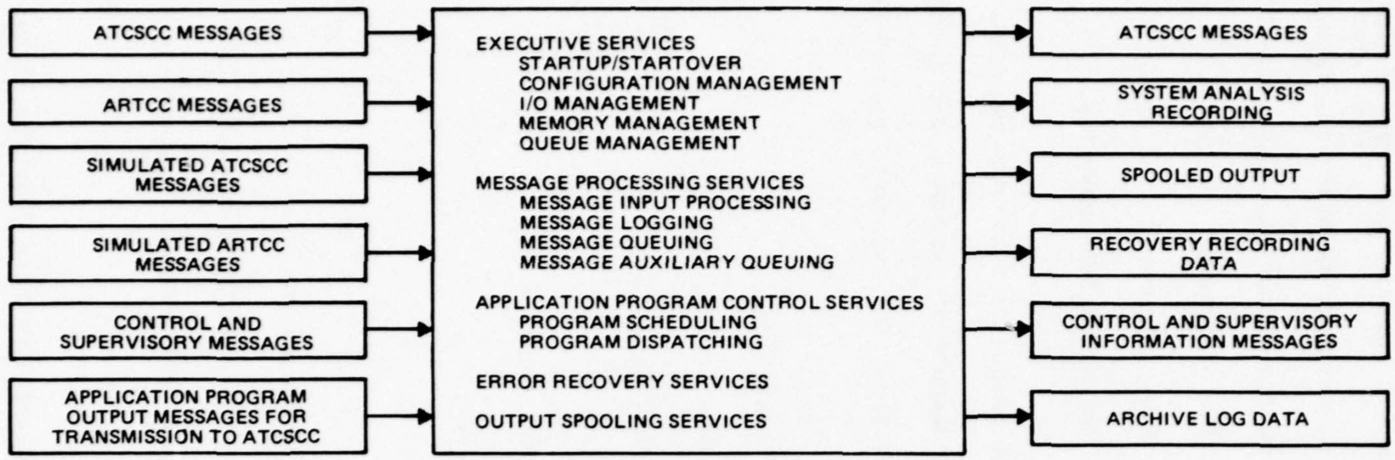


Figure 2.2.1-6. EX-1.0 Executive Subsystem (EX) Overview

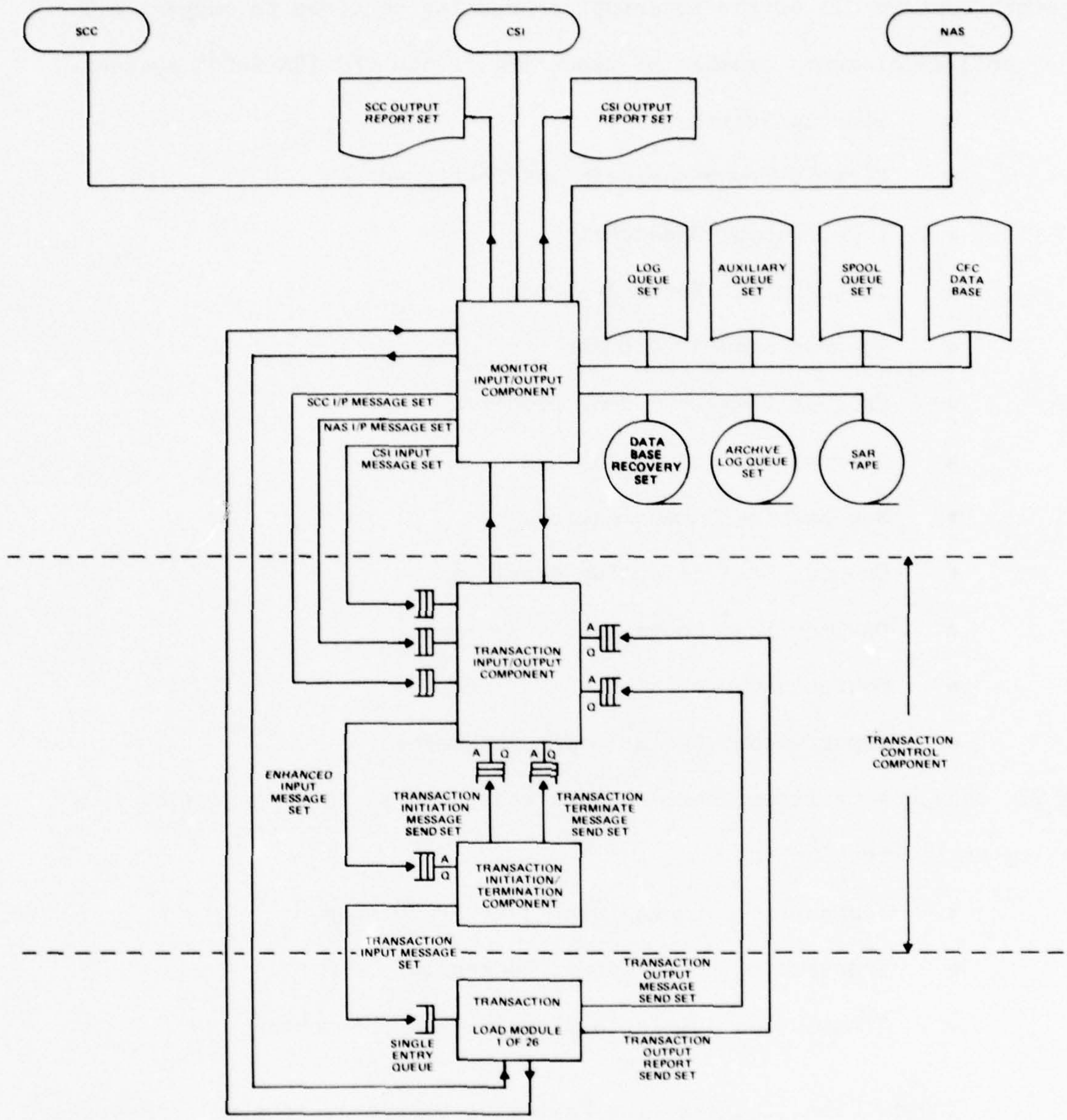


Figure 2.2.1-7. Interaction of the CFC Monitor Component and the Transaction Control Component

together perform all of the interrupt processing required to support all other program elements capable of executing in the CFC IBM 9020A system:

- Startup/Startover
- Element Error Analysis and Configuration
- Input/Output Management
- Input/Output Error Analysis
- Porgram Element Control
- Program Element Synchronization
- Storage and Communications
- Man-Machine Communications
- On-Line Data Recording Services
- On-Line Test Tools
- Contents Supervision
- Input/Output Device Dependent Code

The TCC consists of three subcomponents that control the transaction processing within the System:

- Transaction Startup/Startover Processing
- Transaction Input/Output Processing
- Transaction Initiation/Termination Processing

e. Input. The inputs to the Executive Subsystem consist of messages from the ARTCCs and the ATCSCC as well as operator and supervisory messages entered from the Input/Output Typewriter (IOT) or card reader. The terminal input from the ATCSCC and ARTCCs is processed by the CFC Monitor Component, only to the extent of translating the input data as necessary into the proper internal data representation, removing nonmessage characters, and then queuing the input for the Transaction Control Component. Input from IOT and card-reader

devices are processed by the CFC Monitor Component and responses are prepared and returned as necessary, or the messages are queued to the proper program element for which the message was destined.

f. Output. The outputs from the Executive Subsystem consist of messages to the ATCSCC, System Analysis Recording (SAR) Data, Archive Log Queue Data, Applications Input Message Data, and TA Input Message Data. In addition, responses to any operator or supervisory input messages are output to the proper destination.

2.2.1.2.1 CFC Monitor (EX-MON) Component

The CFC Monitor (EX-MON) Component is a modified version of the NAS En Route Monitor, with the modifications falling into two major categories. The first category consists of modifications to remove modules and COMPOOL tables that pertain to hardware that are not configured for the System, or that pertain to data files that do not exist on the System. To minimize the impact of the modifications, only the minimum code deletions are specified. Any COMPOOL, table references that might cause unresolved external references have been deleted, as were references or calls to nonexistent modules.

The second category of modifications to the NAS En Route Monitor to implement it as the CFC Monitor Component, involves the inclusion of code to support the requirements of the System and to support the ATCSCC and ARTCC terminal processing. These modifications are discussed with each component description overview.

2.2.1.2.1.1 Startup/Startover (EX-SUSO) Component

The Startup/Startover (EX-SUSO) Component is responsible for the initialization of the CFC Monitor Component and Transaction Control Component. The initialization is effected through the performance of an Initial Program Load (IPL), which is a hardware-implemented function. After the operator selects the physical device address of the unit containing the IPL data, he depresses the Load Button. An automatic hardware initiated input loads six words of data into the Preferential Storage Area of a selected Storage Element (SE), and these data cause the loading of the Startup/Startover Control (EXMSS) Module. At the termination of the loading, control is passed to the start of EXMSS, which controls the execution of the remaining modules whose responsibility is to load the remainder of the operational system. Upon completion of the startup of the monitor, control is passed to the Transaction Control Component to enable it to perform TCC startup and then to the Database Management Subsystem to enable it to perform data base startup.

In the event of a malfunction of a hardware component that would prevent further processing due to lost data or a question of data validity, a startover is initiated. The significant difference between startup and startover is that startup is effected by an IPL, whereas startover is effected automatically upon detection of an error. Control is passed to the TCC startover processing software after configuring the malfunctioning hardware component out of the System.

2.2.1.2.1.2 Element Error Analysis (EX-EEA) Component

The Element Error Analysis (EX-EEA) Component analyzes machine errors and

responds with the appropriate actions,and initializes and maintains the hardware configuration. This component has been incorporated into the CFC Monitor Component without modifications.

2.2.1.2.1.3 Input/Output Management (EX-IOM) Component.

The Input/Output Management (EX-IOM) Component consists of a series of routines that initiate the physical input and output operations through the privileged instructions,and supervise the resulting operations. Because the references to physical devices are contained in the Monitor COMPOOL Set (EXMCS), no modifications were required to the EX-IOM Component to support the ATCSCC and ARTCC devices.

2.2.1.2.1.4 Input/Output Error Analysis (EX-IOE) Component

The Input/Output Error Analysis (EX-IOE) Component is called by the Input/Output Management (EX-IOM) Component whenever an error is detected. Each error is analyzed by the appropriate module and corrective action is taken to retry the operation or notify the originator of the error condition. This component has been included in the CFC Monitor Component with table modifications only.

2.2.1.2.1.5 Program Element Control (EX-PEC) Component

The Program Element Control (EX-PEC) Component dispatches Program Elements (PEs) on Compute Elements (CEs) or Input/Output Compute Elements (IOCEs); processes supervisor call (SVC) and external interrupts; maintains the date and the time; controls elapsed processing and I/O time; moves data between Storage Elements (SEs) and Maintenance and Channel (MACH) Storage; and

schedules, loads, and terminates PEs. This component has been modified to initialize and maintain the epoch time, to dispatch PEs in main storage to IOCEs (as well as to CEs), and to provide other processing necessary to implement a full multiprocessing environment.

2.2.1.2.1.6 Program Element Synchronization (EX-PES) Component

The Program Element Synchronization (EX-PES) Component is responsible for synchronizing and resolving resource contention between Compute Elements (CEs) and Input/Output Compute Elements (IOCEs) and executing Program Elements (PEs). When a PE needs to utilize resources or data, the EX-PEC Component provides the means to ensure that other PEs do not attempt to utilize the same resource or modify data until the first PE is finished. This component has been incorporated into the CFC Monitor Component without modifications.

2.2.1.2.1.7 Storage and Communications (EX-SAC) Component

The Storage and Communications (EX-SAC) Component is responsible for the servicing of requests for pooled storage resources and for the queue and software communications services. Any PE requiring temporary ownership of memory space, internal communications paths between PE input queues, or logical devices makes its request through the EX-SAC Component. Also, the placing of data blocks in a queue, or the locating, retrieving, or deleting of data blocks from a queue is accomplished through calls to the EX-SAC Component.

2.2.1.2.1.8 Man-Machine Communications (EX-MMC) Component

The Man-Machine Communications (EX-MMC) Component provides the input and output

interface between operations and supervisory personnel and the CFC Software System. The input and output can be initiated and returned to either an IBM 1052 Input/Output Typewriter (IOT) or an IBM 2540 Card Reader. The input is accepted by the EX-MMC Component, formatted by use of Monitor COMPOOL Set (EXMCS) tables, and sent to the logical device specified. This component was modified to accommodate any operator or supervisor communications required by other components of the OPCX. The modifications consisted mainly of placing appropriate data into the proper COMPOOL Tables and inserting the message codes into EX-MMC internal tables.

2.2.1.2.1.9 Online Data Recording (EX-ODR) Component

The Online Data Recording (EX-ODR) Component provides the capability to record data on the System Analysis Recording (SAR) tape. The data to be recorded on the tape may be specified by the originator of the data and is routed to the SAR tape through specific Supervisor Calls (SVCs). This component has been implemented in the CFC Monitor Component with modifications only for deletion of references to nonexistent COMPOOL tables or devices.

2.2.1.2.1.10 Online Test Tools (EX-OLT) Component

The Online Test Tools (EX-OLT) Component provides the capabilities for simulation tape input, online loading, program interrupt and abort processing, and dump generation. Of particular importance is the program interrupt, as this processing is involved in the implementation of the Input/Output Compute Element (IOCE) dispatching of Program Elements. The EX-OLT Component required extensive modifications to implement the IOCE dispatching capability, because a PE that is dispatched on an IOCE and subsequently issues an instruction not valid on an IOCE should be checked for re-dispatch on a Compute Element rather

than aborted. Because an IOCE has neither floating point nor decimal arithmetic capability, the attempted execution of these instruction types will cause a program interrupt. The IOCE program interrupt module EXQPX will be modified to requeue the PE for dispatch as "CE-only executable." Thus, it would subsequently be dispatched on a CE only. A further modification to EXQPX has been made to check for the presence of a PE interrupt control block, which contains the location of logic within the PE to process the interrupt. This capability exists for PEs executing under control of a CE and must be implemented in the IOCE interrupt module, because the PE may be dispatched on either processor type.

2.2.1.2.1.11 Contents Supervision (EX-CTS) Component

The Contents Supervision (EX-CTS) Component supports the dynamic buffering and online edit capability for the loading of non-main-storage-resident Program Elements (PEs). This component has been implemented in the CFC Monitor Component with no modifications except the deletions of references to deleted COMPOOL tables.

2.2.1.2.1.12 Input/Output Device Dependent Code (EX-DDC) Component

The Input/Output Device Dependent Code (EX-DDC) Component consists of the modules responsible for the input and output processing for configured devices. The component is I/O interrupt driven and, upon the occurrence of an interrupt, the proper module will be called to process the data. This component has been modified to permit the recognition of the ARTCC and ATCSCC terminals, to perform the input and output processing for these devices, and to store and retrieve the data in the proper queues for the Transaction Control Component.

2.2.1.2.2 Transaction Control (EX-TCC) Component

a. Purpose. The Transaction Control Component (TCC) logically extends the functions of the CFC Monitor Component to include a startup and startover processing unique to the CFC Software System, transaction message logging, auxiliary queuing for in-core message queues, transaction output report spooling, transaction initiation/termination processing, and a simulation input interface. The extended executive functions are performed by adding eleven new program elements OPEs).

b. Function Description. Virtually all functions performed upon or for Application Subsystem transaction load modules are done by the Transaction Control Component. The EX-TCC Component serves as a buffer layer of executive software to isolate the Application Subsystem from a dependence upon the CFC Monitor Component. Although the Transaction Control Component is more dependent upon the CFC Monitor Component, the interface is isolated to a limited set of Supervisor Calls (SVCs) and COMPOOL tables. In some cases, however, the existing SVCs and COMPOOL tables require minor changes to accommodate the EX-TCC interface. By restricting the EX-TCC Component/CFC Monitor Component interface, much of the reliability and documentation of the CFC Monitor Component is preserved, and the Application Subsystem is more transportable to a monitor redesign.

Since the Transaction Control Component is simply a logical name given to the components and modules that extend the monitor functions for the Executive Subsystem, a further functional description is given in terms of the Transaction Control Component's major components: the Transaction Startup and Startover (EX-TSSC) Component, the Transaction Input/Output (EX-TIOC) Component, and the Transaction Initiation/Termination (EX-TITC) Component.

c. Inputs. The TCC processes messages from five sources: SCC input messages, En Route input messages, CSI input messages, transaction output immediate messages, transaction output report messages. Each of these sets is described below.

- SCC Input Messages: Messages originating at the Air Traffic Control System Command Center in Washington,D.C., are referred to as SCC messages. These messages are converted by the CFC Monitor Component to EBCDIC format with all device control characters removed. The monitor places converted SCC messages into LEASEd message blocks, and SENDs the messages to the TCC.

- NAS Input Messages: Messages originating at the Air Route Traffic Control Centers (ARTCCs) that have been blocked and forwarded to the System by the ARTCC store-and-forward computers are referred to as NAS messages. The processing of the NAS messages by the CFC Monitor Component is similar to that of the SCC messages previously described.

- CSI Input Messages: For simulation purposes, SCC and NAS messages can be entered either at the CFC simulated-input (CSI) terminal or by the Online Test Director (TA) Component program element. Control messages to the TA program can also be entered via the CSI terminal. The simulated SCC and NAS messages as well as the TA control messages are referred to as CSI messages. The processing of CSI messages

by the CFC Monitor Component is similar to the processing of the SCC messages previously described.

- Transaction Output Immediate Messages: Output messages from the Application Subsystem Transaction Load Modules are placed in EBCDIC format into LEASEd message blocks and sent to the TCC.

- Transaction Output Report Messages: Output reports from Transaction Load Modules are blocked and placed in EBCDIC format into LEASEd message blocks and sent to the TCC.

d. Outputs. Transaction Control Component processing produces the following output: SCC output immediate messages, SCC output report messages, En Route output immediate messages, CSC output immediate messages, CSI output report messages, transaction input messages, and the Archive Log Queue Set. Each output set is described below.

- SCC Output Immediate Messages: OPCX's acceptance or rejection of an SCC input message as well as informative and progress information relating to the input message are called SCC output messages. The LEASEd message blocks that contain these SCC output messages are sent to the CFC Monitor Component to perform the actual I/O necessary to send the message back to the SCC terminal.

- SCC Output Report Messages: The report responses of the Application Subsystem Transaction Load Modules to the SCC's

request for information are called SCC output reports. These SCC output reports may optionally be sent to many output devices as specified in the transaction control header of the LEASEd message blocks that contain the reports.

- En Route Output Immediate Messages: OPCX's acceptance or rejection of a NAS input message is similar to the processing of the SCC output messages.

- CSI Output Immediate Messages: OPCX's acceptance or rejection of simulated SCC and NAS messages as well as informative and progress information about the simulated messages are called CSI output messages. Instead of sending these output messages to SCC or NAS output devices, the fact that the messages were simulated is recognized and the output messages are sent to the CSI output device.

- CSI Output Report Messages: The report responses to a simulated SCC request for information are called CSI output reports. These reports are recognized to be the result of simulated SCC messages and the actual I/O to send these reports to the SCC report output device is not performed. Instead these reports are sent to the CSI output report device.

- Transaction Input Messages: Each logical input message is enhanced by the TCC by placing a transaction control

header into a LEASEd message block which contains the input message. These message blocks are then sent to the appropriate Application Subsystem Transaction Load Module to process that type of message.

- Archive Log Queue Set. Each transaction input message entering the OPCX is logged by the TCC on the archive log queue tape. In addition to logging the input messages, the TCC logs enhanced messages, initiation messages, output messages, data base statistics messages, and termination messages. The archive log queue tapes serve as a record of the CFC transactions processed by the OPCX. The set of archive log queue tapes forms the Archive Log Queue Set.

2.2.1.2.2.1 Transaction Startup and Startover (EX-TSSC) Component

a. Purpose. The purpose of the Transaction Startup and Startover (EX-TSSC) Component is to provide initialization of CFC-unique resources during System startup and to provide reinitialization of those same resources during System startover. The EX-TSSC Component consists of a startup component and a startover component that are invoked by the Startup/Startover Management Component of the CFC Monitor Component.

b. Functional Description. There are two subcomponents that make up the Transaction Startup and Startover Component: the Startup Component (EX-SUC) and the Startover Component (EX-SOC). Functionally, the EX-SUC and EXSOC Components are the startup and startover PEs invoked by the CFC Monitor Component.

During System startup an Initial Program Load (IPL) procedure is performed by the CFC Monitor Component to make the OPCX operational. A series of startup routines are then sequentially invoked to perform specific initialization functions. The initialization functions often have predecessor dependencies that necessitate the sequential invocations. The EX-SUC Component assumes a fully operational CFC Monitor Component and therefore is invoked last by the monitor during startup.

The CFC-unique resources that are initialized by the EX-SUC Component include: TCC control tables, queue sets, and tape data sets. There are three subcomponents of the Startup Component to perform the specific initializations: the Initialize Data Area (EX-IDAC) Component, the Initialize Disk Data Sets (EX-IDDC) Component, and the Initialize Tape Data Sets (EX-ITDC) Component.

The EX-IDAC Component performs an execution-time adaptation of the CFC-unique TCC control tables. These tables include the following sets:

- Transaction Property Set
- Transaction Initiation/Termination Control Set
- File Property Set
- Optional Output Device Set
- Queue Control Set
- PE Property Set
- Physical Device Property Set
- Transaction Control Component Global Data Set

The EX-IDDC Component initializes the following data sets: Log Queue Set,

Auxiliary Queue Set, Spool Queue Set. The disk data sets are used for the queue sets in the System.

The EX-ITDC Component initializes the Archive Log Queue Set. This tape data set may span physical tape volumes.

Startover processing by the CFC Monitor Component involves a sequential invocation of initialization routines. Like startup initialization, the initialization functions often have predecessor dependencies that necessitate the sequential invocations. The EX-SOC Component assumes a fully operational CFC Monitor Component and therefore is invoked last by the monitor during startover.

The EX-SOC Component is divided into two subfunctions: reinitializing TCC control tables, and input message startover. The components that perform these subfunctions are the Reinitialize Data Area (EX-RDAC) Component and the Input Message Startover (EX-IMSC) Component. The EX-RDAC Component performs the reinitialization of the TCC control tables.

The EX-IMSC Component is responsible for the startover of input messages. While operational, the EX-TCC Component logs all transaction input messages, enhanced input messages, transaction initiation messages, transaction output immediate messages, and transaction termination messages into the Log Queue Set. The state of transaction input messages at the time of startover can be easily ascertained by interrogating the data hierarchy of the Log Queue Set. The data hierarchy of the Log Queue Set is described in Section 3.

The input messages that were initiated when startover occurred are terminated by the EX-TTPC Subcomponent of the EX-IMSC Component. Messages that were

received but not yet initiated when startover occurred are reinstated.

2.2.1.2.2.2 Transaction Input/Output (EX-TIOC) Component

a. Purpose. The purpose of the Transaction Input/Output (EX-TIOC) Component is to provide three input/output functions not performed by the CFC Monitor Component. The three functions include: transaction logging, auxiliary queuing, and transaction output report spooling. The EX-TIOC Component consists of eight dispatchable program elements (PEs).

b. Functional Description. There are three subcomponents that make up the Transaction Input/Output Component: the Log Queue (EX-LQC) Component, the Auxiliary Queue (EX-AQC) Component, and the Spool Queue (EX-SQC) Component. Though the functions performed by the three subcomponents are somewhat unrelated, they do share queue management as a common function. Queue management provides a means of forming a data hierarchy on physical devices that support random access. The queue sets associated with the EX-TIOC Component (i.e., the Log Queue Set, the Auxiliary Queue Set, and the Spool Queue Set) are initialized during the EX-TSSC Component startup processing. The structure of the data hierarchy unique to each queue set is established by EX-LQC, EX-AQC, and EX-SQC Components through calls to the common queue management primatives: EXGFQM, EXRFQM, EXQRDM, and EXQWTM.

Transaction Input Messages, Enhanced Input Messages, Transaction Initiation Messages, Transaction Output Immediate Messages, and Transaction Termination Messages are logged into the Log Queue Set by the four subcomponents of the

EX-LQC Component: Input Log Queue (EX-ILQC) Component, Initiation Message Log (EX-IMLC) Component, Output Message Log (EX-OMLC) Component, and Termination Message Log (EX-TMLC) Component. These subcomponents perform the message logging by "pulling" log queue records from the Free Queue Entry Pool and chaining them into the appropriate level of the Log Queue Set data structure.

Logging of transaction messages serves two purposes in the CFC Executive Subsystem. The Log Queue Set hierarchy provides a quick means of determining the state of all transaction input messages during system startover and for archiving the transaction messages to the Archive Log Queue Set. The Archive Log Queue Set is a tape used by the Support Complex to do system performance evaluation and data reduction.

The NAS En Route Monitor, which forms the baseline for the CFC Monitor Component, supports an in-core message block queue that is the means of inter-PE communication. The message block queue is a system resource that often is the cause of process blockage; the size of the PE queue is limited by total system core requirements. The Ex-AQC Subcomponent of the EX-TIOC Component interfaces with the CFC Monitor Component to provide a logical extension of the in-core message blocks. The logical extension, referred to as auxiliary queuing, interfaces through the monitor SVCs associated with message block management (i.e., SEND, DELETE, ACCEPT,...etc). Messages sent to full in-core messages queues are placed into the Auxiliary Queue Set by the SEND SVC. The SEND SVC also sets a flag in the Auxiliary Queue Control

Set indicating that all future SENDs must reroute messages destined to that queue (this will preserve the first in, first out (FIFO) nature of the in-core message queue). PEs always attempt to obtain messages from an in-core queue first by issuing an ACCEPT SVC. If there are no in-core messages the EX-AQDC Component attempts to dequeue a message block from the Auxiliary Queue Set and place it on the appropriate in-core queue. When no more messages are found in the Auxiliary Queue Set, the EX-AQDC Component turns off the flag in the Auxiliary Queue Control Set previously set by the EX-AQEC Component. When the flag is turned off, the SEND SVC can continue to SEND messages directly to the in-core message queue.

Transaction Output Report Messages destined to go to common onsite or SCC pointers are spooled to the Spool Queue Set. Output spooling allows Applicable Subsystem transaction load modules (TLMs) to output report messages without having to SEIZE an output device queue. The data structure of the Spool Queue Set is used to designate the ownership of the spooled output reports by input message ID. The implementation of output report spooling requires no interface with the CFC Monitor Component. The TLMs SEND output reports directly to the Spool Input (EX-SINC) Component, which is a subcomponent of the EX-SQC Component. The EX-SINC Component performs the report spooling by creating, when necessary, entries at the Spool ID level of the spool queue hierarchy and entering the output reports reflecting their ownership. When a TLM terminates, a Transaction Termination Message is sent, via SEND, to the Output Message Log (EX-OMLC) Component who SENDS it to

the Spool Input (EX-SINC) Component by the Transaction Initiation/Termination Component. The Transaction Termination Message indicates to the EX-SINC Component that the spooled output reports for the terminated TLM may be scheduled for output to the designated output devices. The EX-SINC Component then SENDs the Transaction Termination Message on to the Spool Output (EX-SOTC) Component. The EX-SOTC Component unspools the reports and SENDs them to the CFC Monitor Component to perform the actual output.

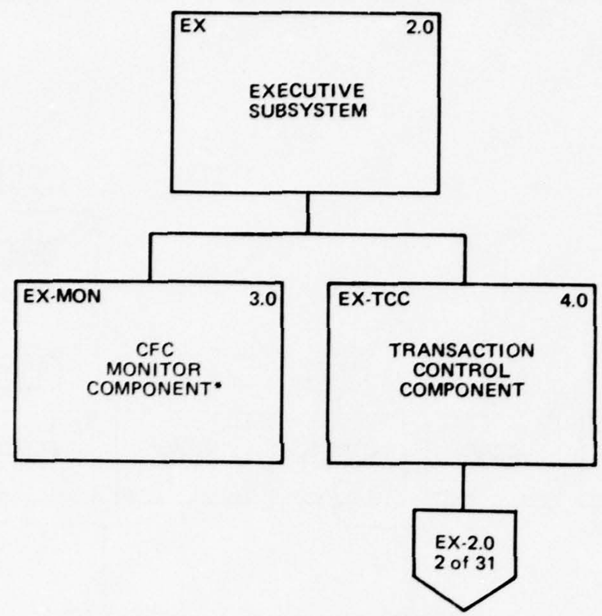
2.2.1.2.2.3 Transaction Initiation/Termination (EX-TITC) Component

a. Purpose. The purpose of the Transaction Initiation/Termination (EX-TITC) Component is to perform the initiation of the appropriate Application Subsystem transaction load modules (TLMs); to SEND Transaction Initiation Messages; and to SEND Transaction Termination Messages.

b. Functional Description. Application Subsystem TLMs receive no unsolicited inputs from the CFC Monitor Component. TLMs are made dispatchable only when the Transaction Initiation/Termination Component SENDs an enhanced input message to the Application Input Message Set. Once an application PE is initiated, the EX-TITC Component CEASEs its own execution until the TLM terminates. Because of the one-to-one relationship between the EX-TITC Component and a TLM, more than one EX-TITC Component must be dispatchable to have more than one TLM dispatchable. The Transaction Initiation/Termination Component is reenterable and is, therefore, multiply dispatchable.

Enhanced Input Messages are ACCEPTed by the EX-TITC Component and the transaction type is compared to those found in the Transaction Property Set. Once a match has been found, a conditional RESERVe is issued for the message block queue of each TLM PE ID capable of processing that type message. Because, by design, each TLM has a single entry input queue, if the RESERVe is successful, the TLM is available for dispatching. A Transaction Initiation Message is sent, via SEND, to be logged, and the Enhanced Input Message is SENDed to the TLM indicating that the TLM is to be dispatched. The EX-TITC Component then CEASEs its own execution to wait for the termination of the TLM. Upon termination of the TLM, the EX-TITC Component SENDs a Transaction Termination Message to the Output Message Log Component.

The Visual Table of Contents for the EX Subsystem is shown in Figure 2.2.1-8.



*THE CFC MONITOR COMPONENT IS GOVERNMENT FURNISHED SOFTWARE AND MODULE HIERARCHY INFORMATION IS NOT EXPANDED HERE. SEE (REF. FOR MONITOR SDD) FOR AN EXPLANATION AT THE MONITOR STRUCTURE.

Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (1 of 31)

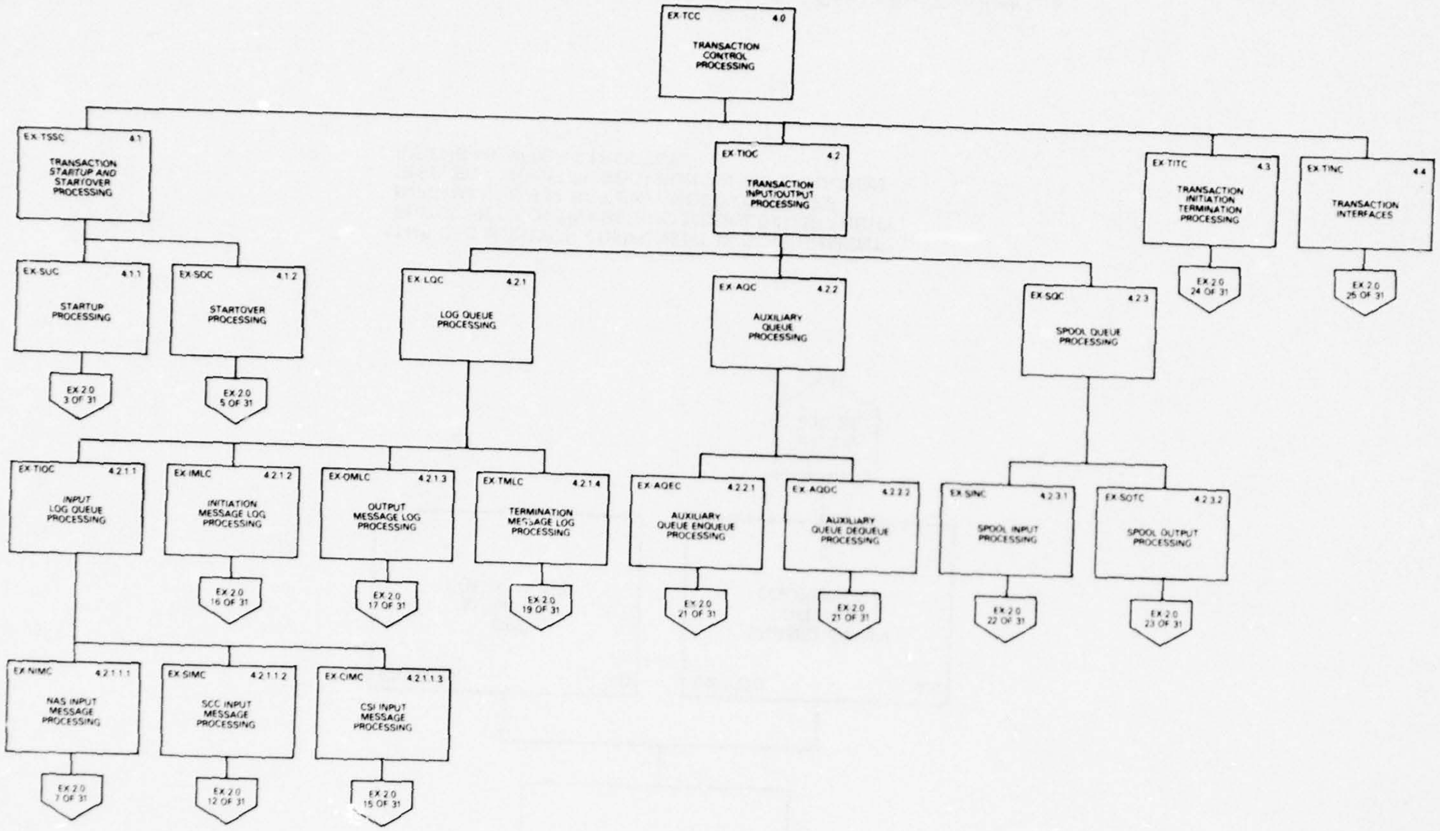


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (2 of 31)

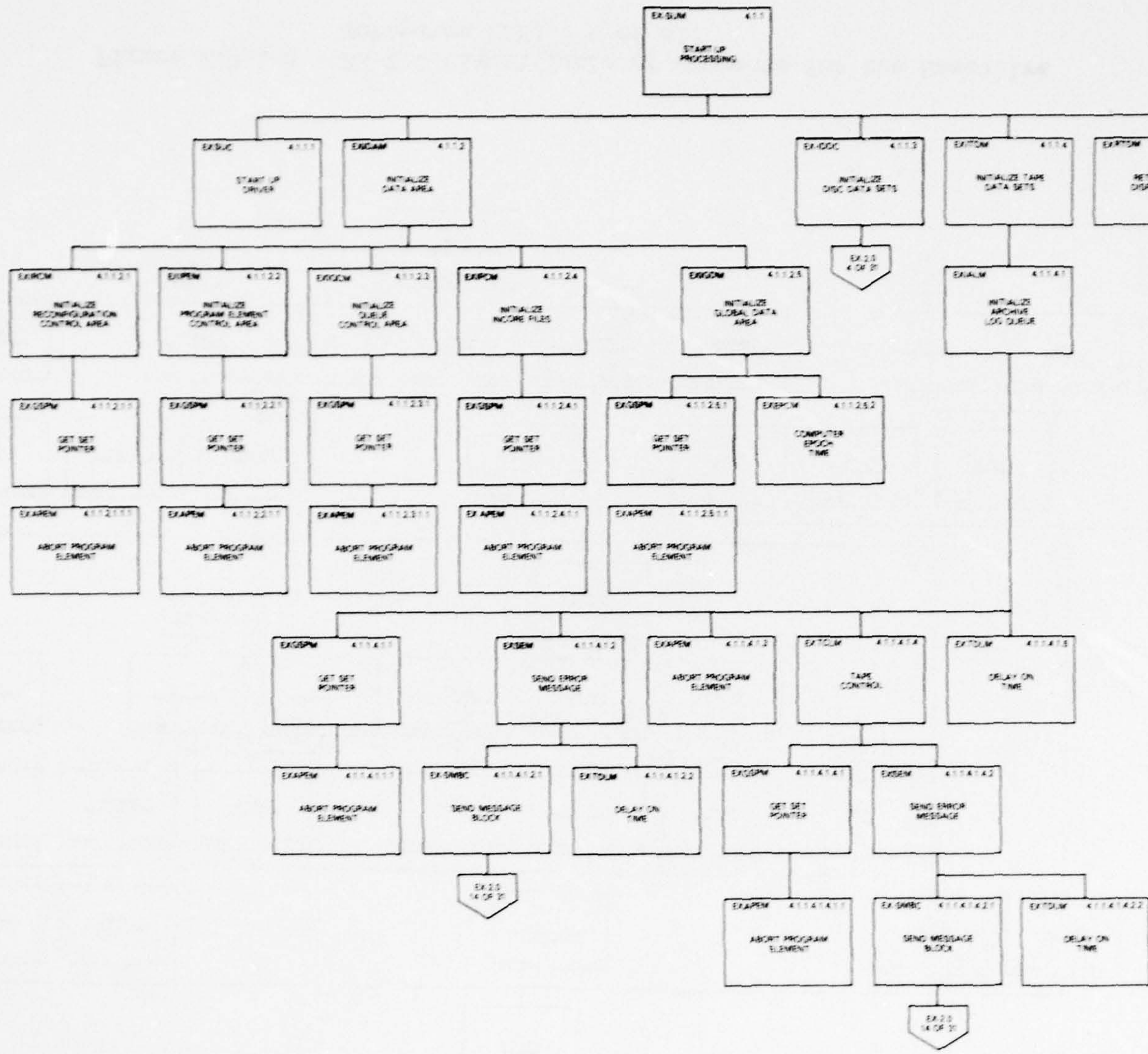


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (3 of 31)

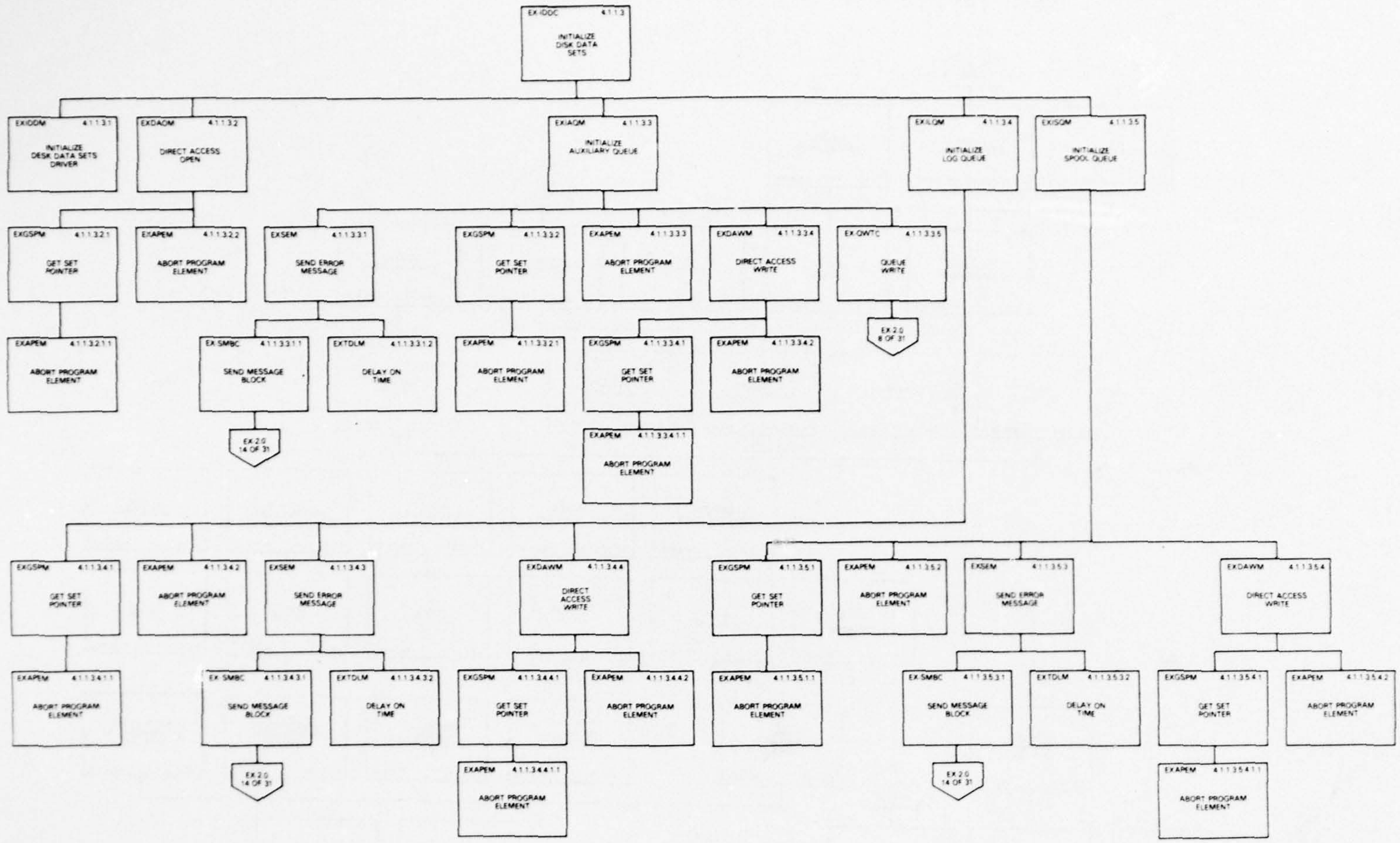


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (4 of 31)

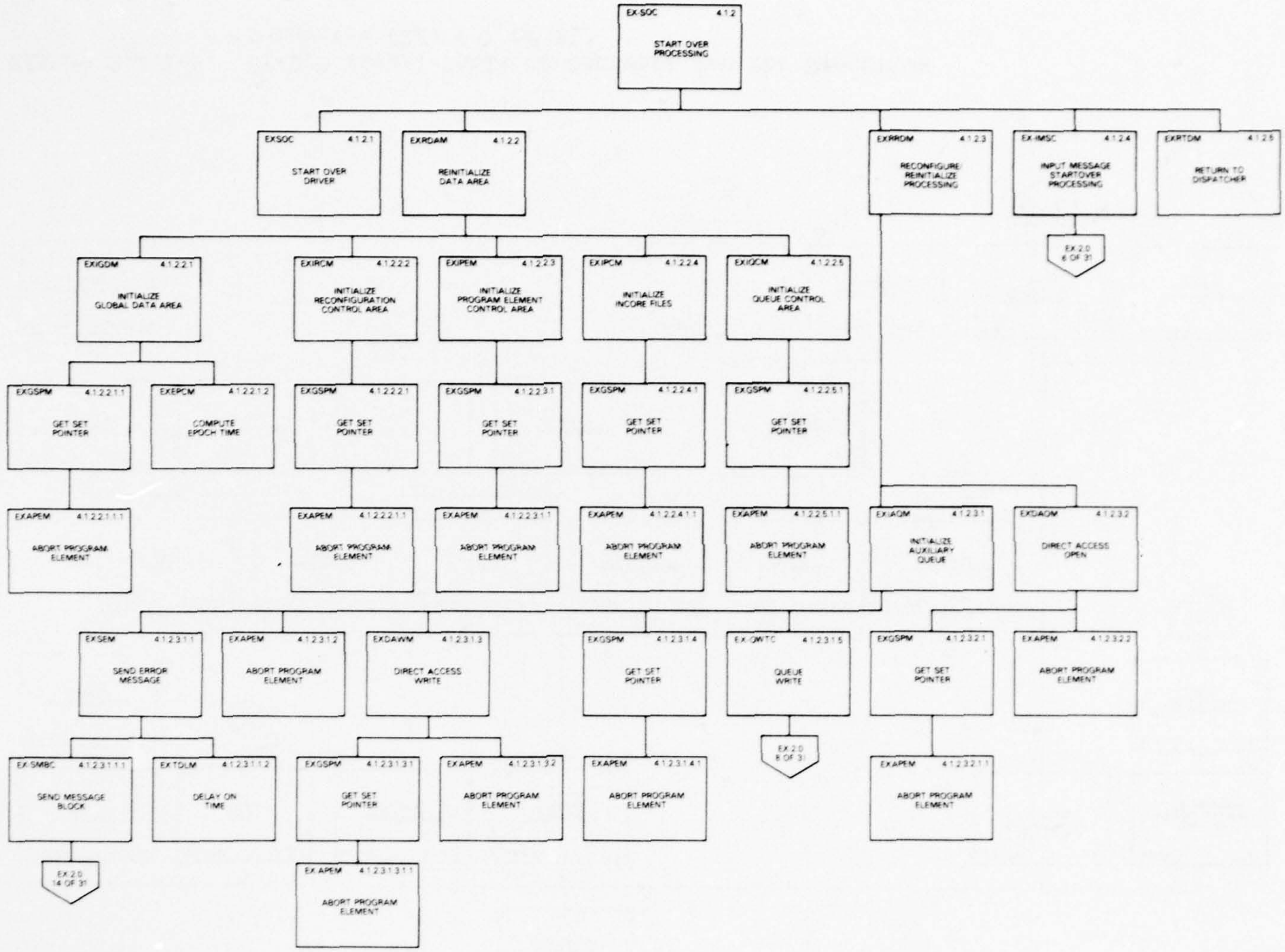


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (5 of 31)

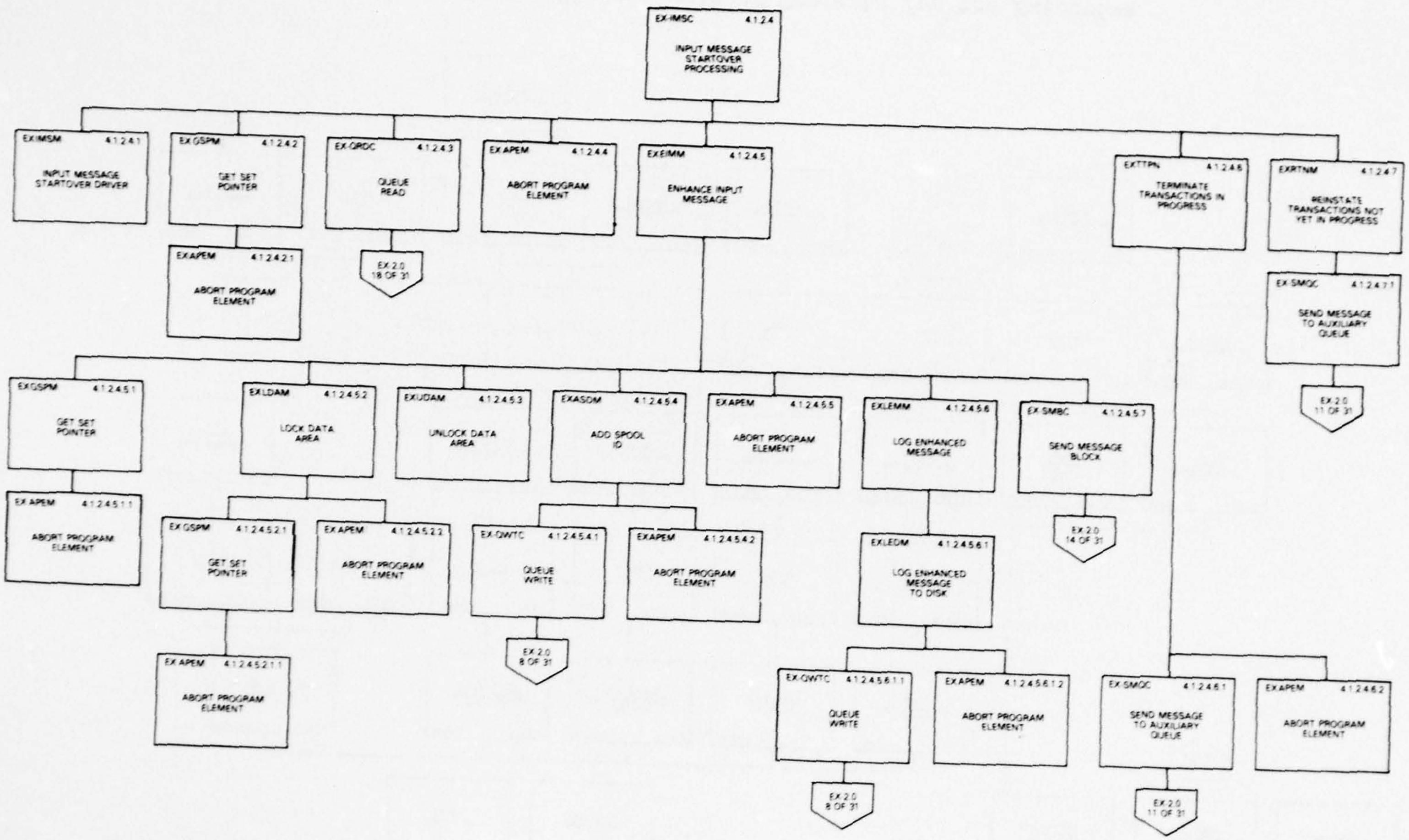


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (6 of 31)

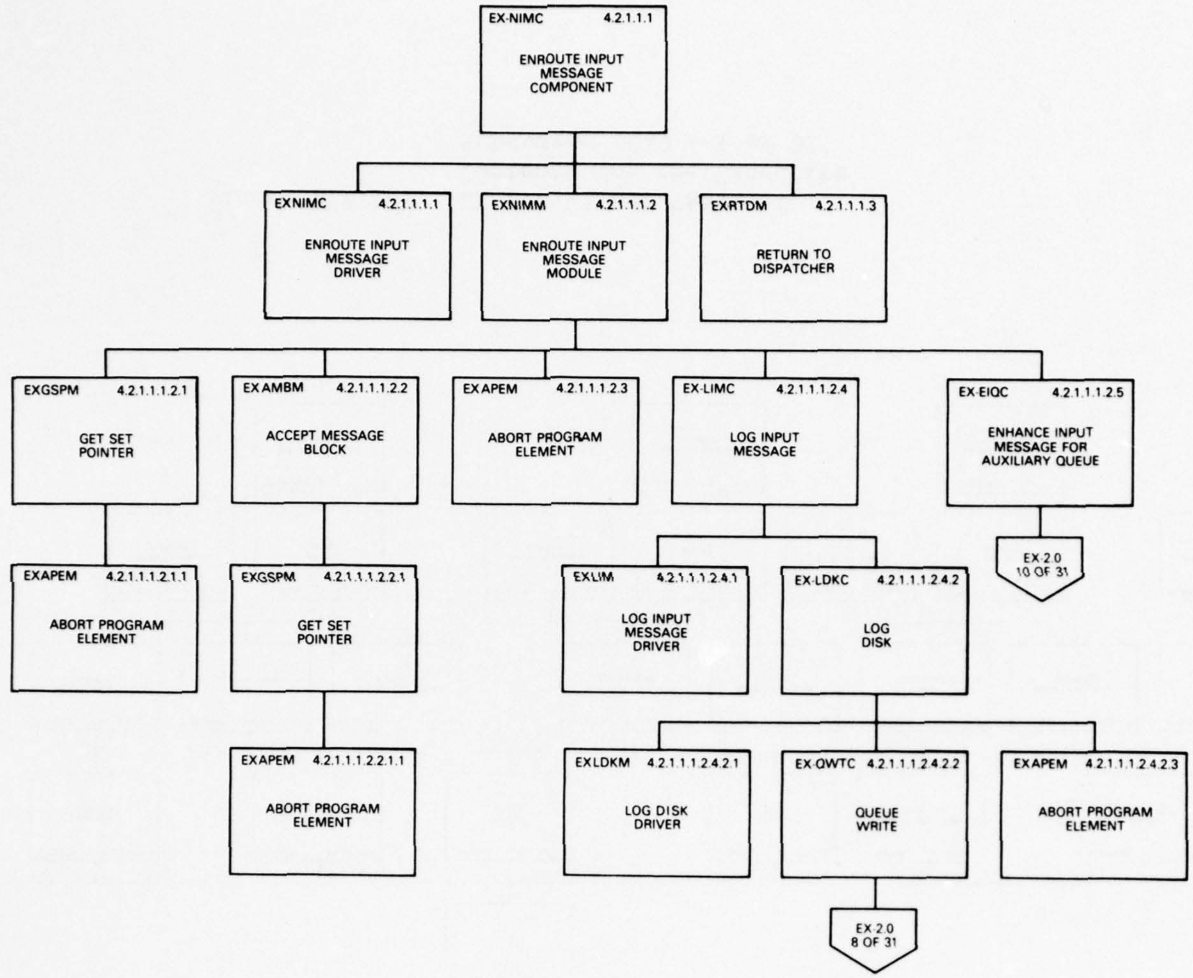


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (7 of 31)

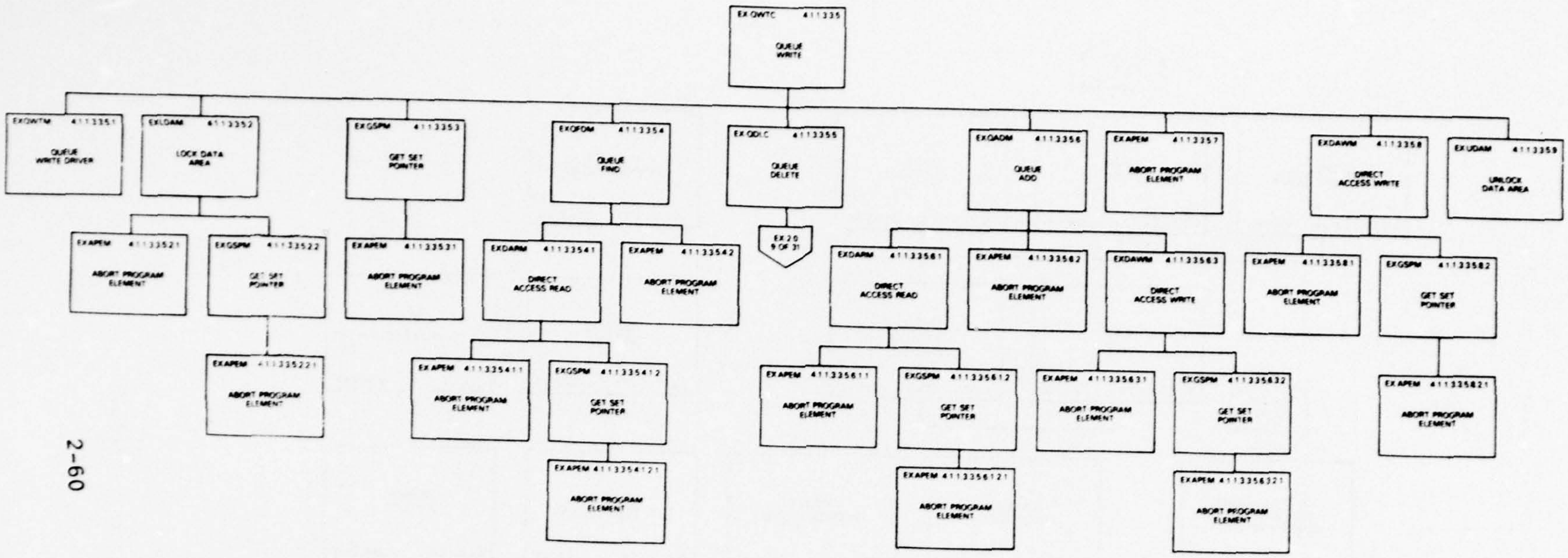


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (8 of 31)

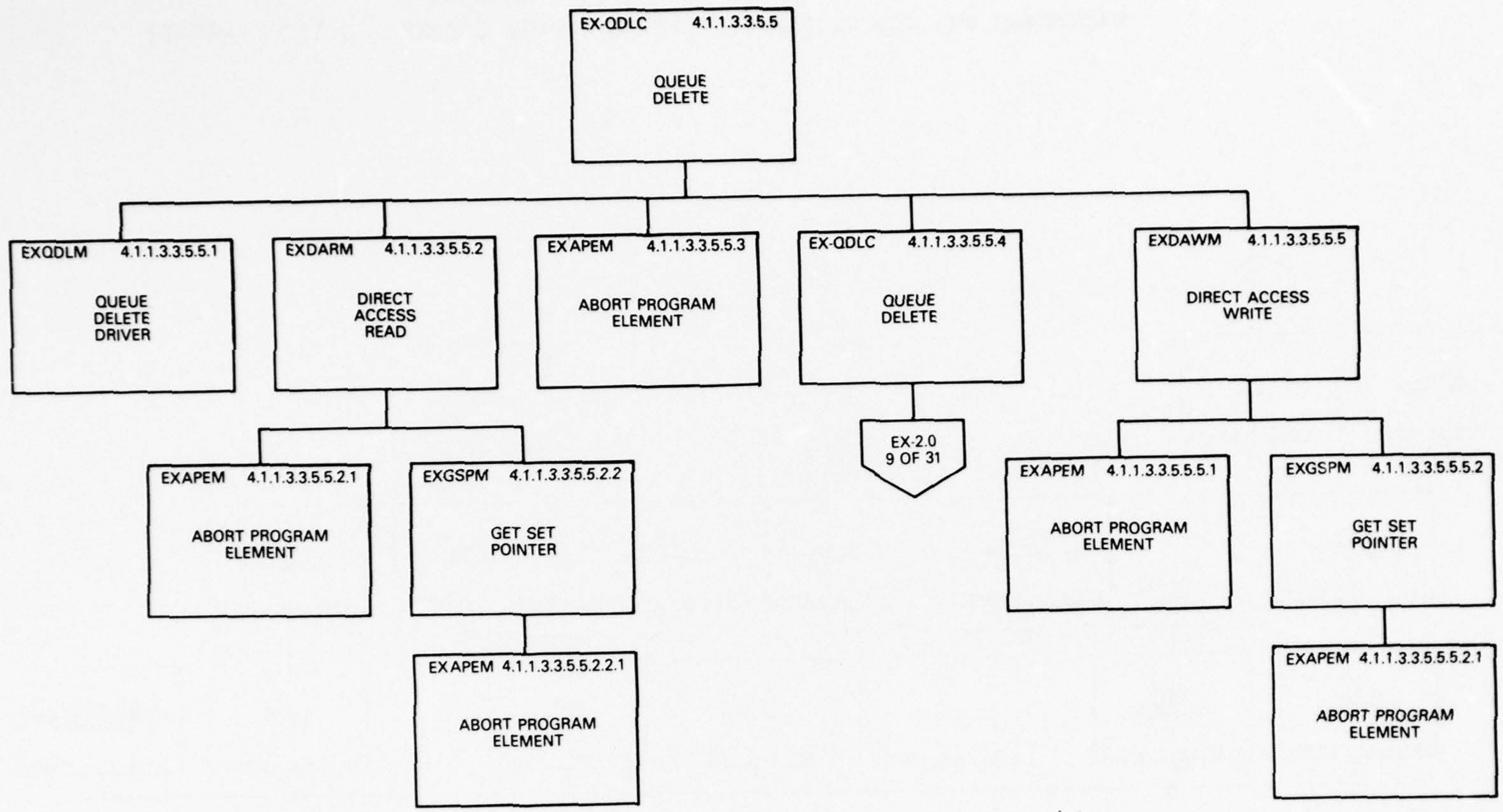


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (9 of 31)

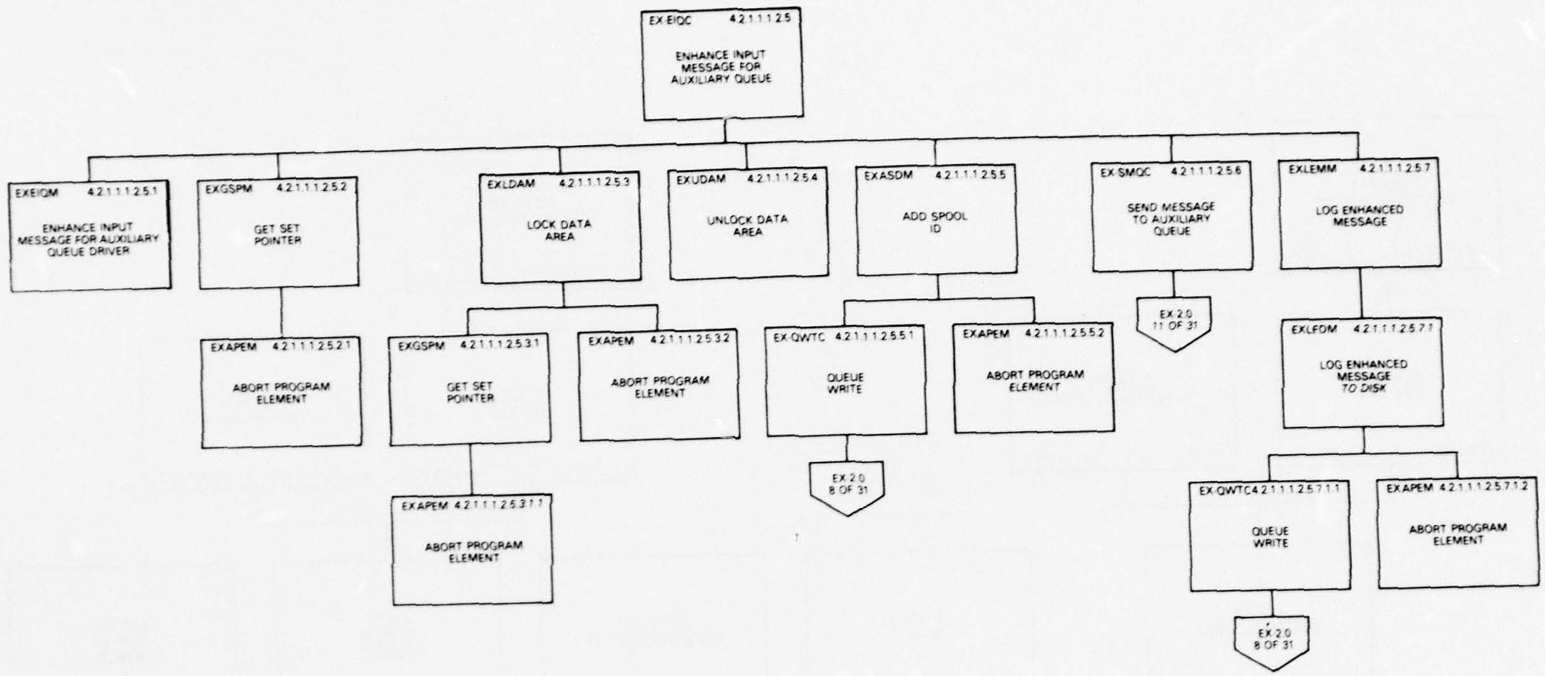


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (10 of 31)

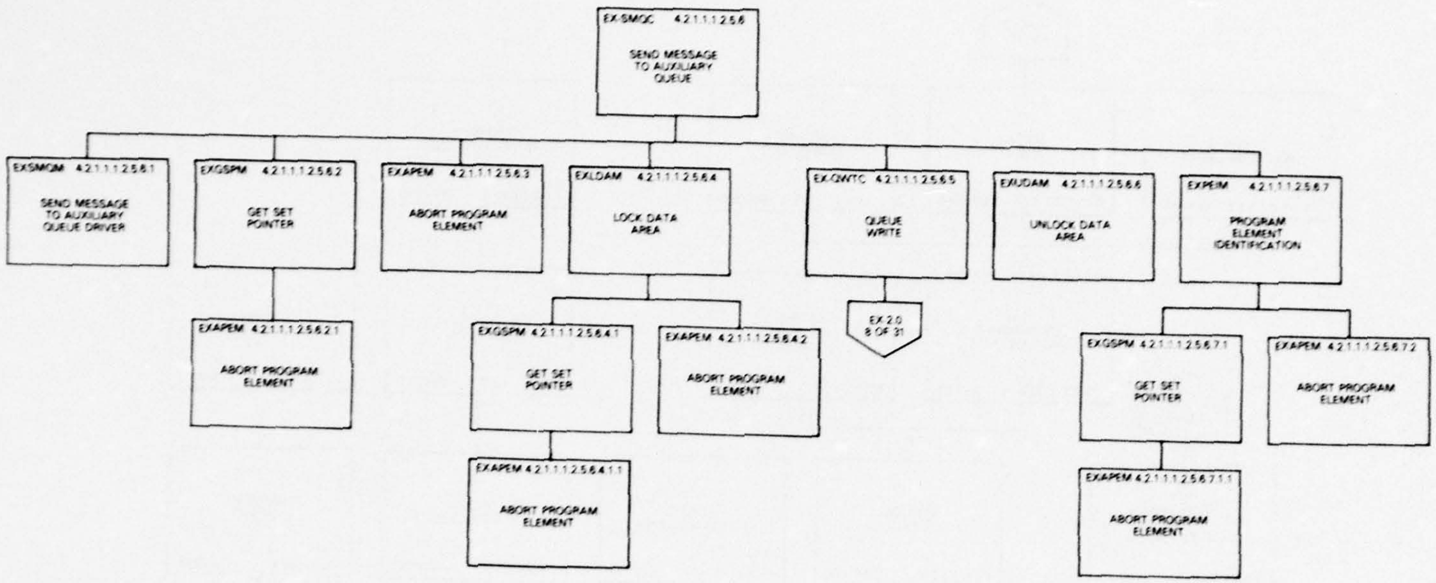


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (11 of 31)

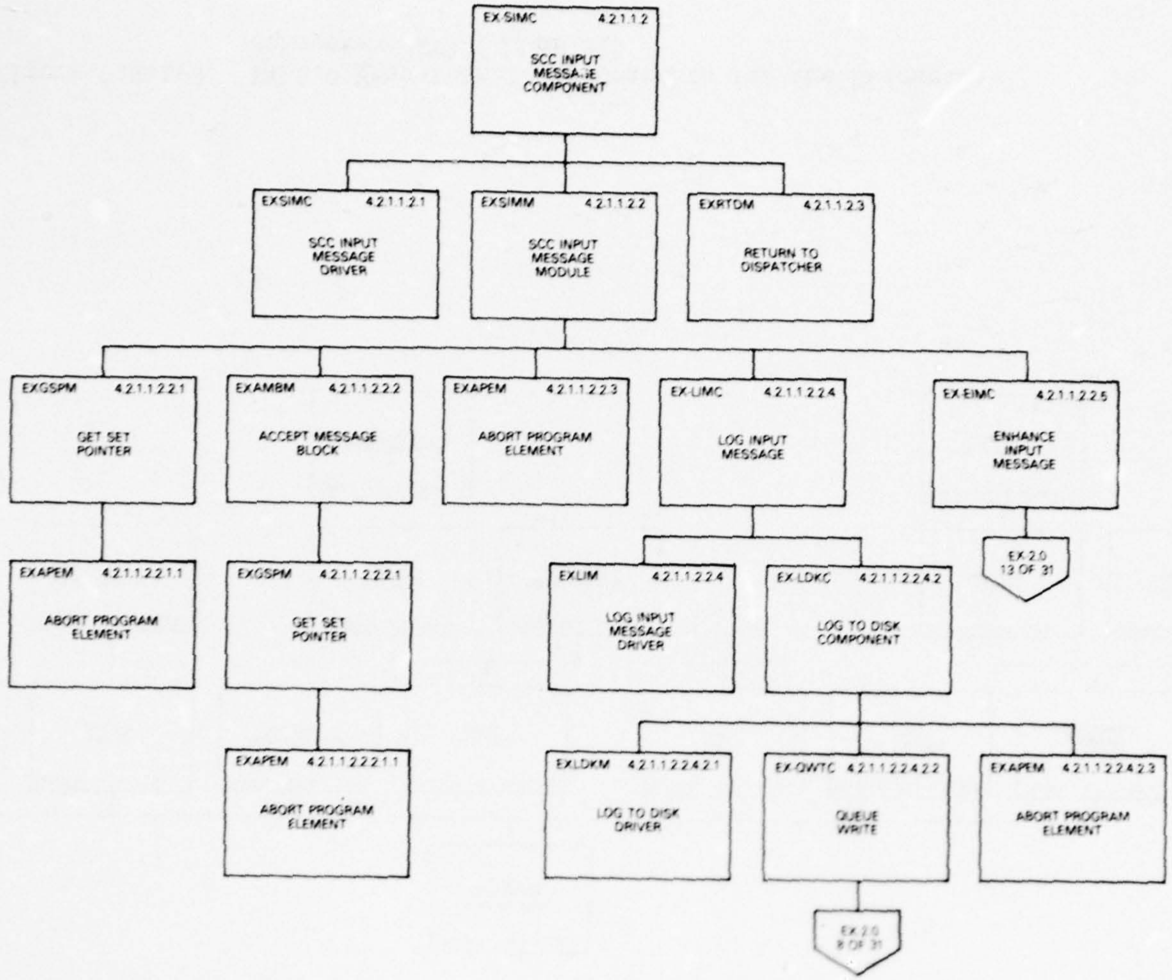


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (12 of 31)

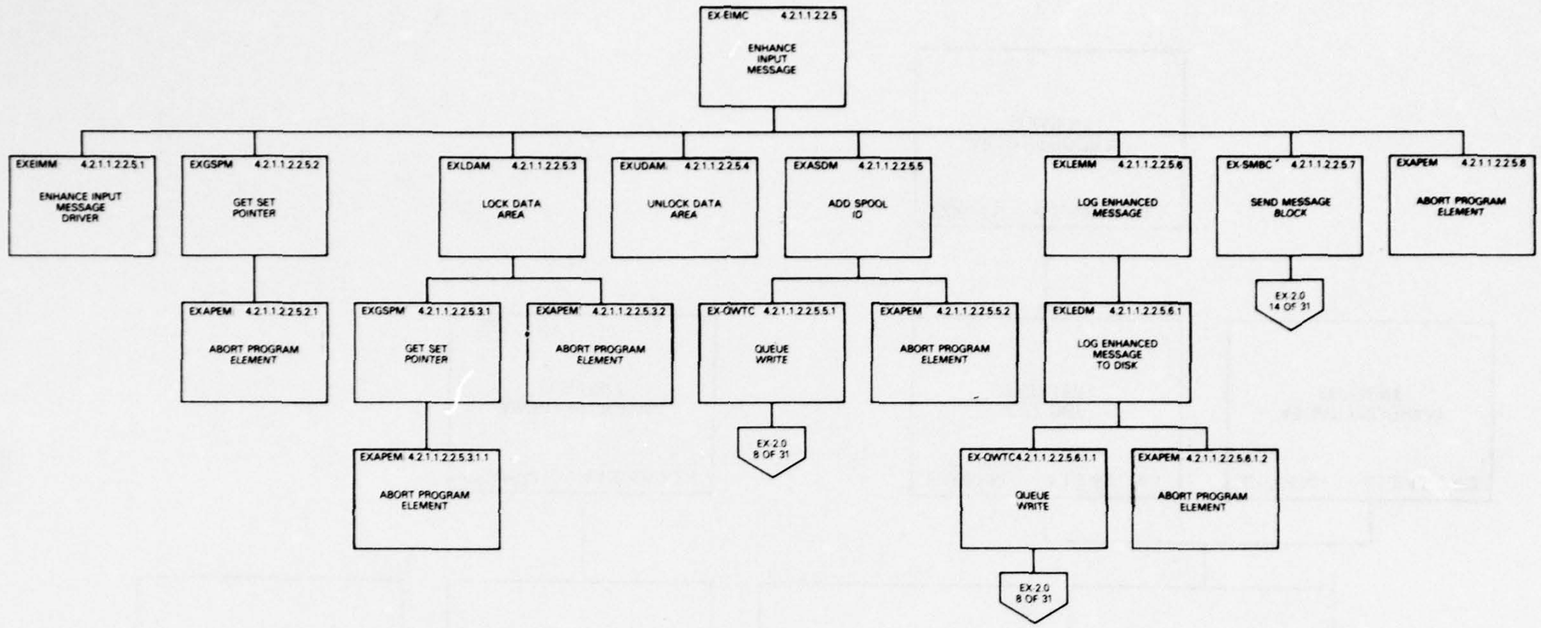


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (13 of 31)

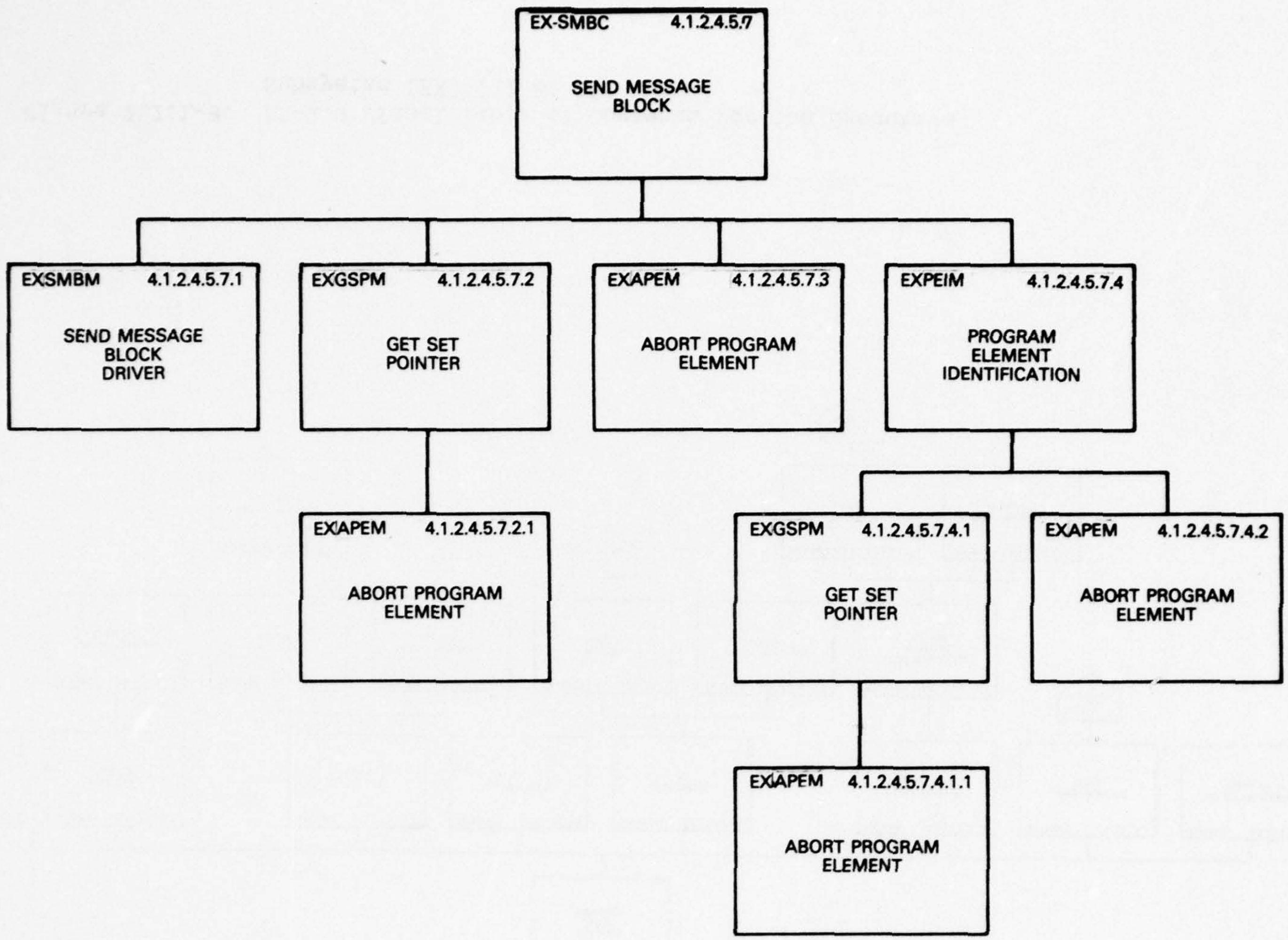


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (14 of 31)

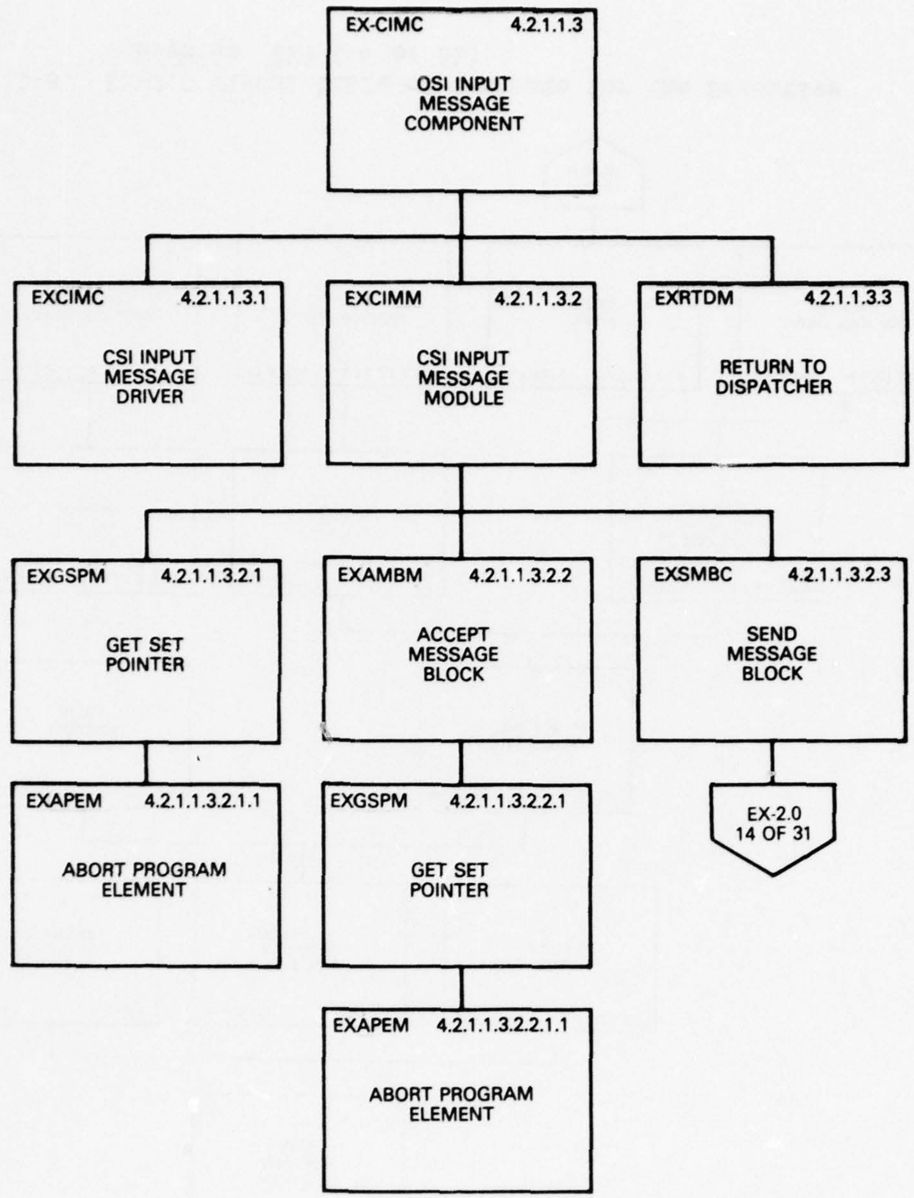


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (15 of 31)

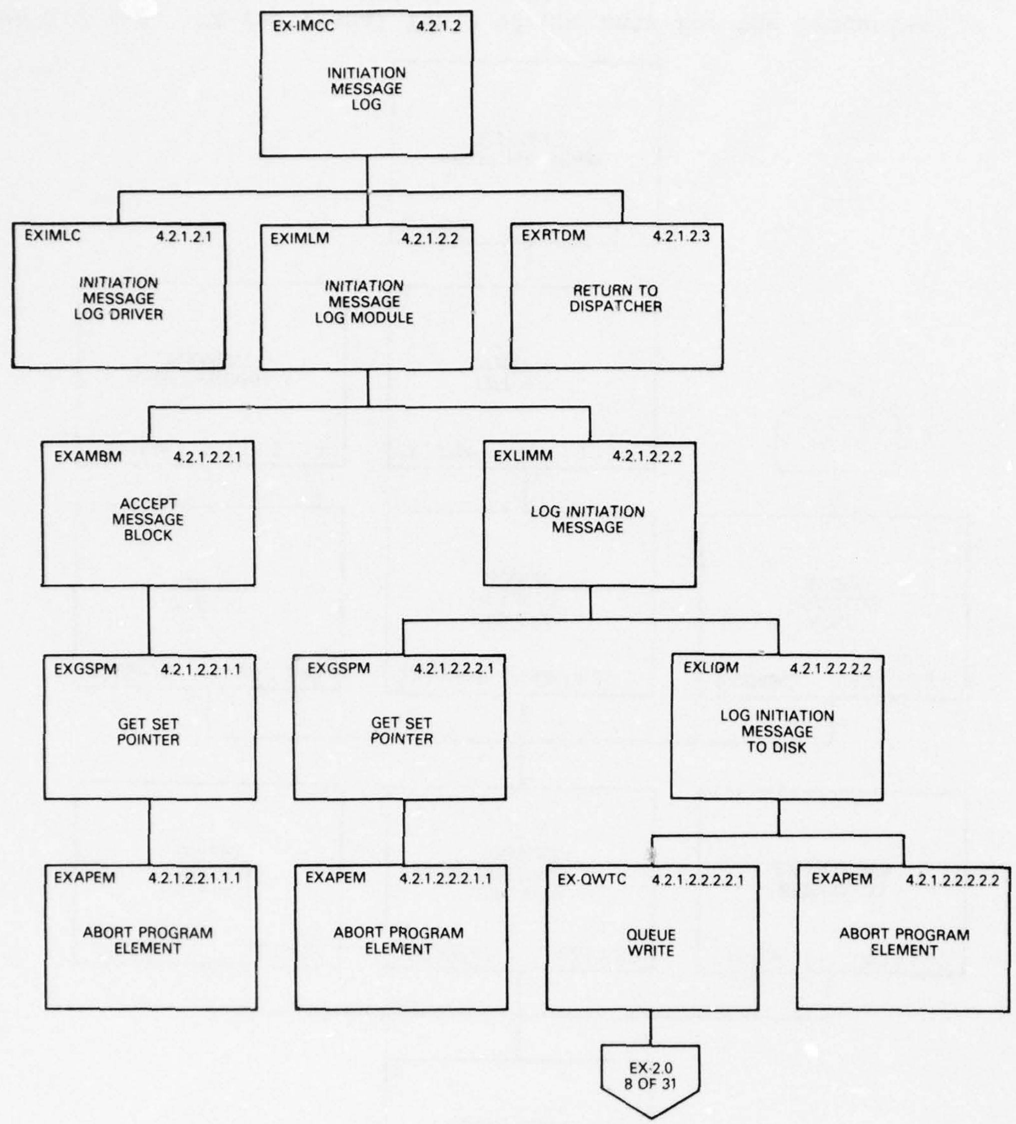
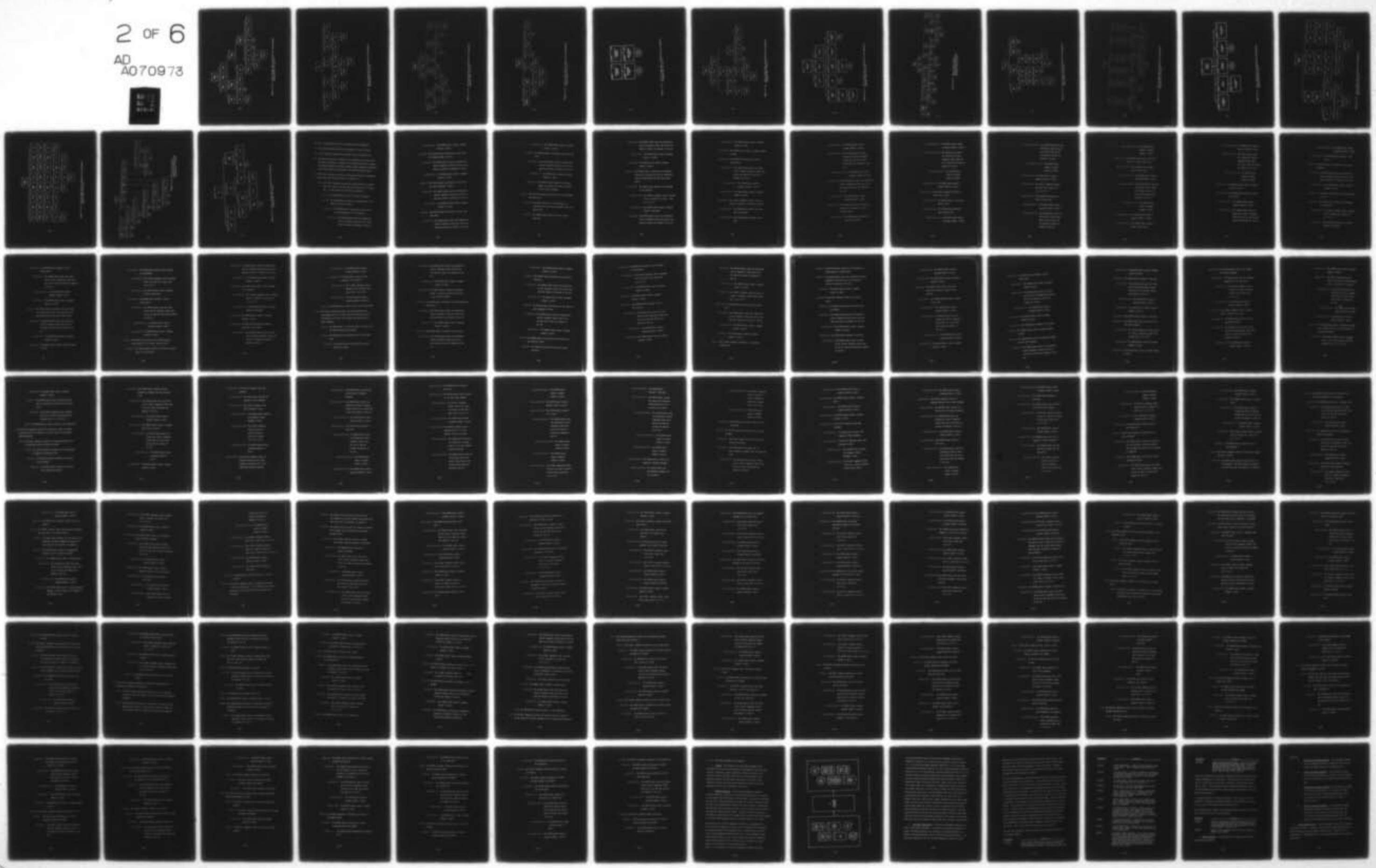
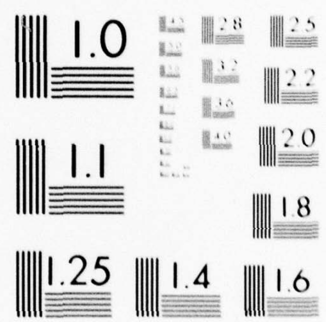


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (16 of 31)

AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955
UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

2 OF 6
AD
A070973





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

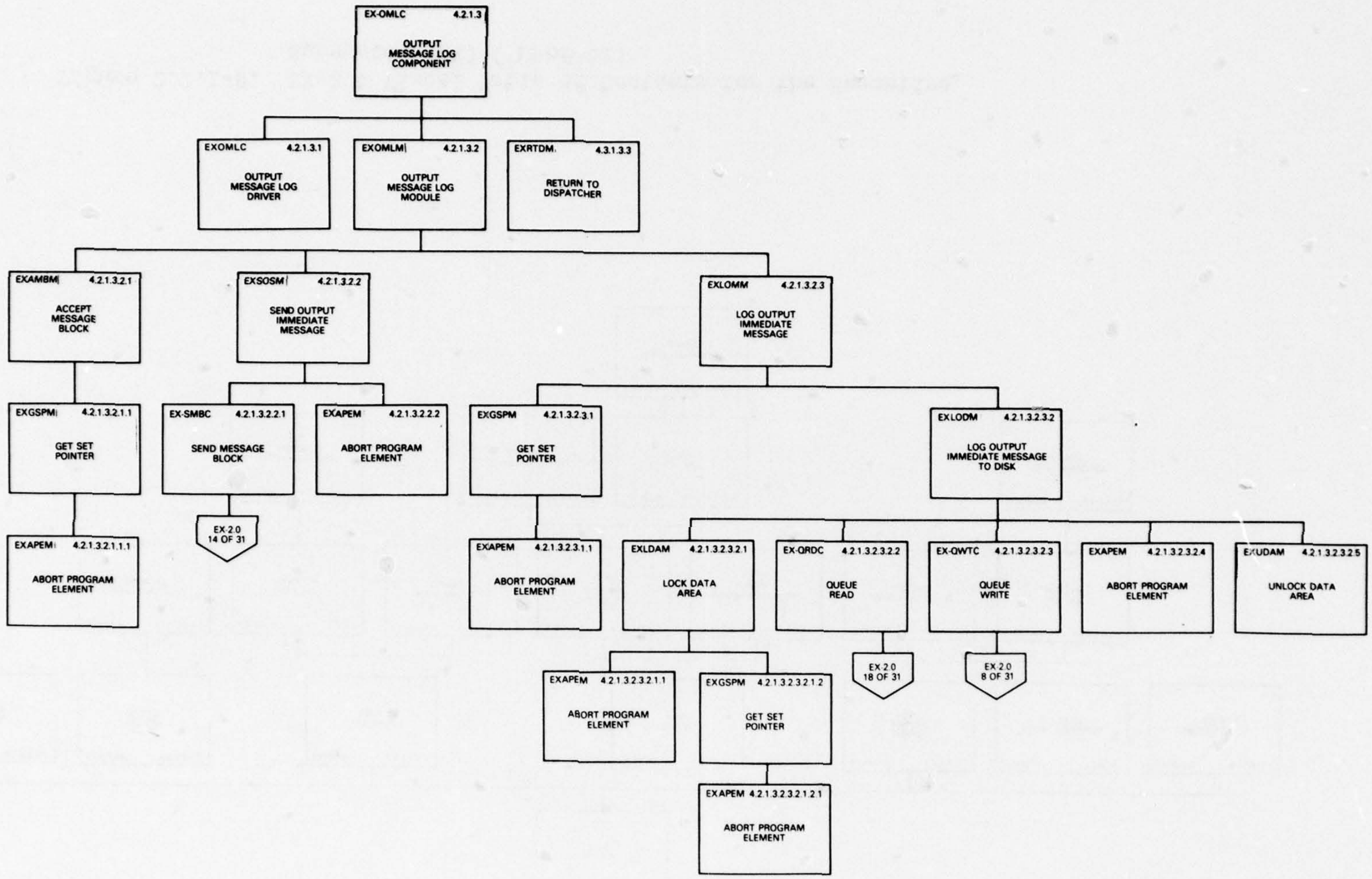


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (17 of 31)

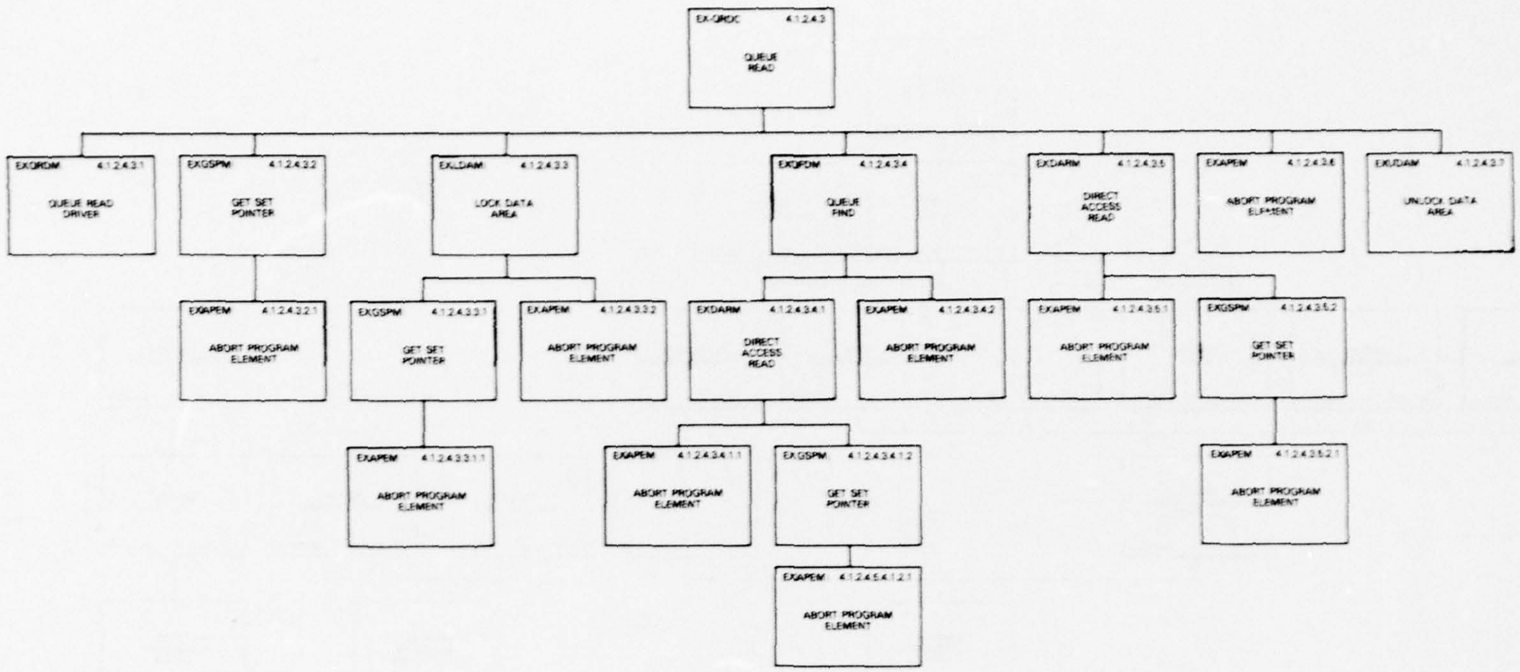


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (18 of 31)

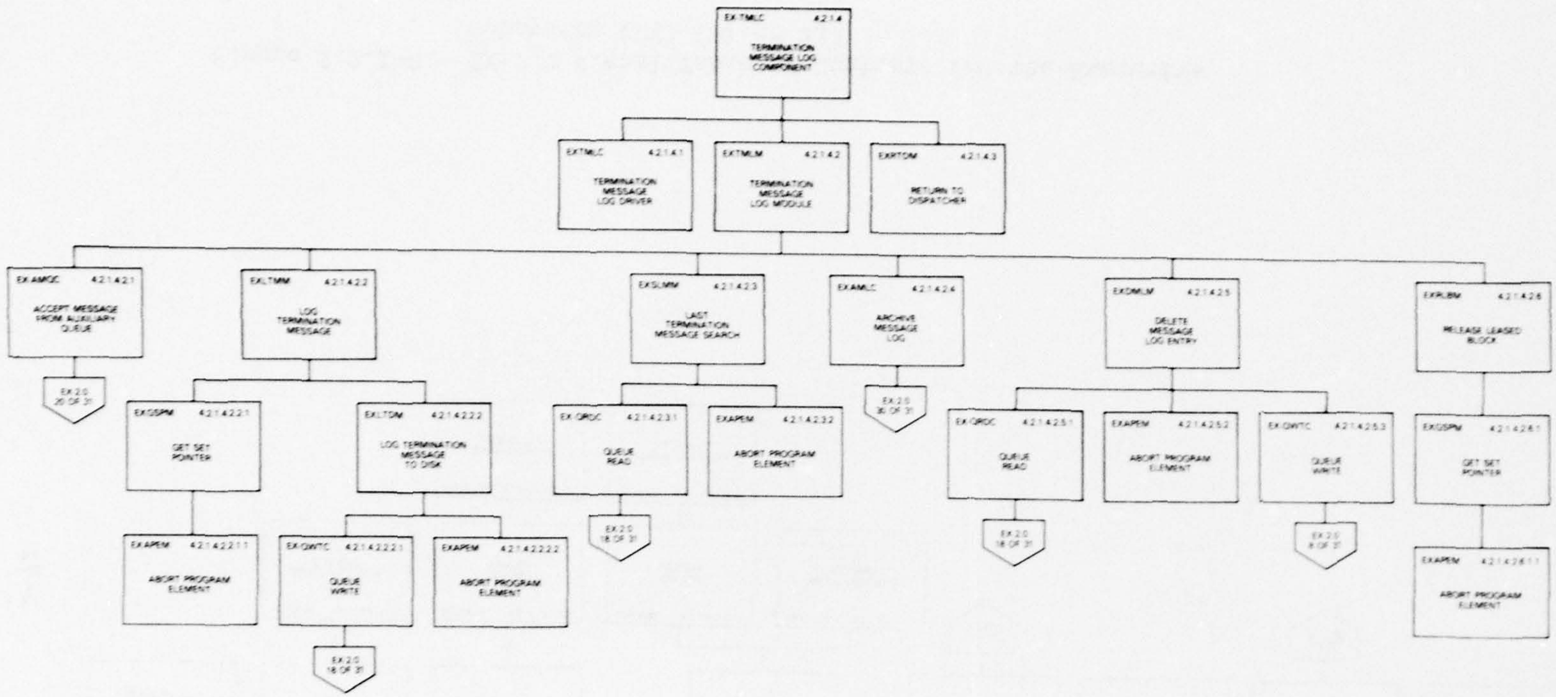


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (19 of 31)

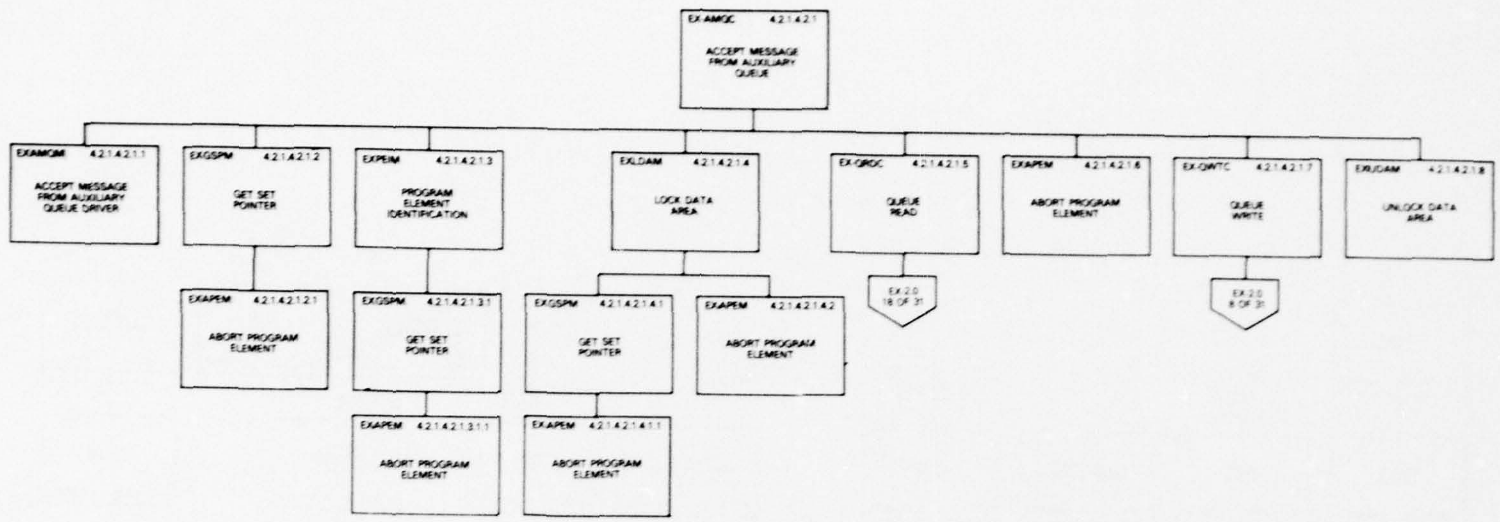


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (20 of 31)

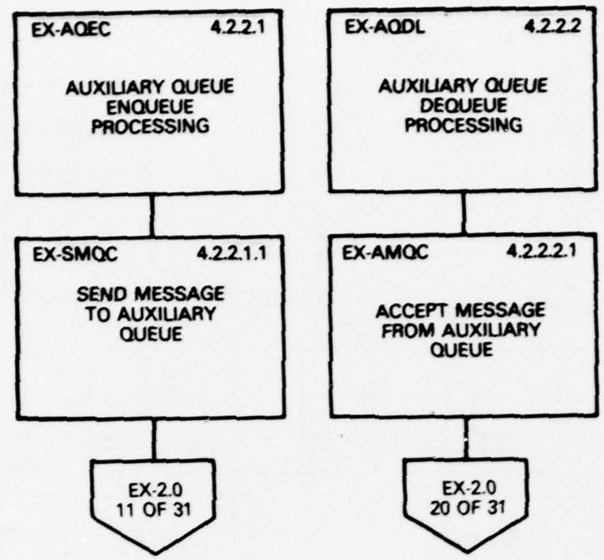


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (21 of 31)

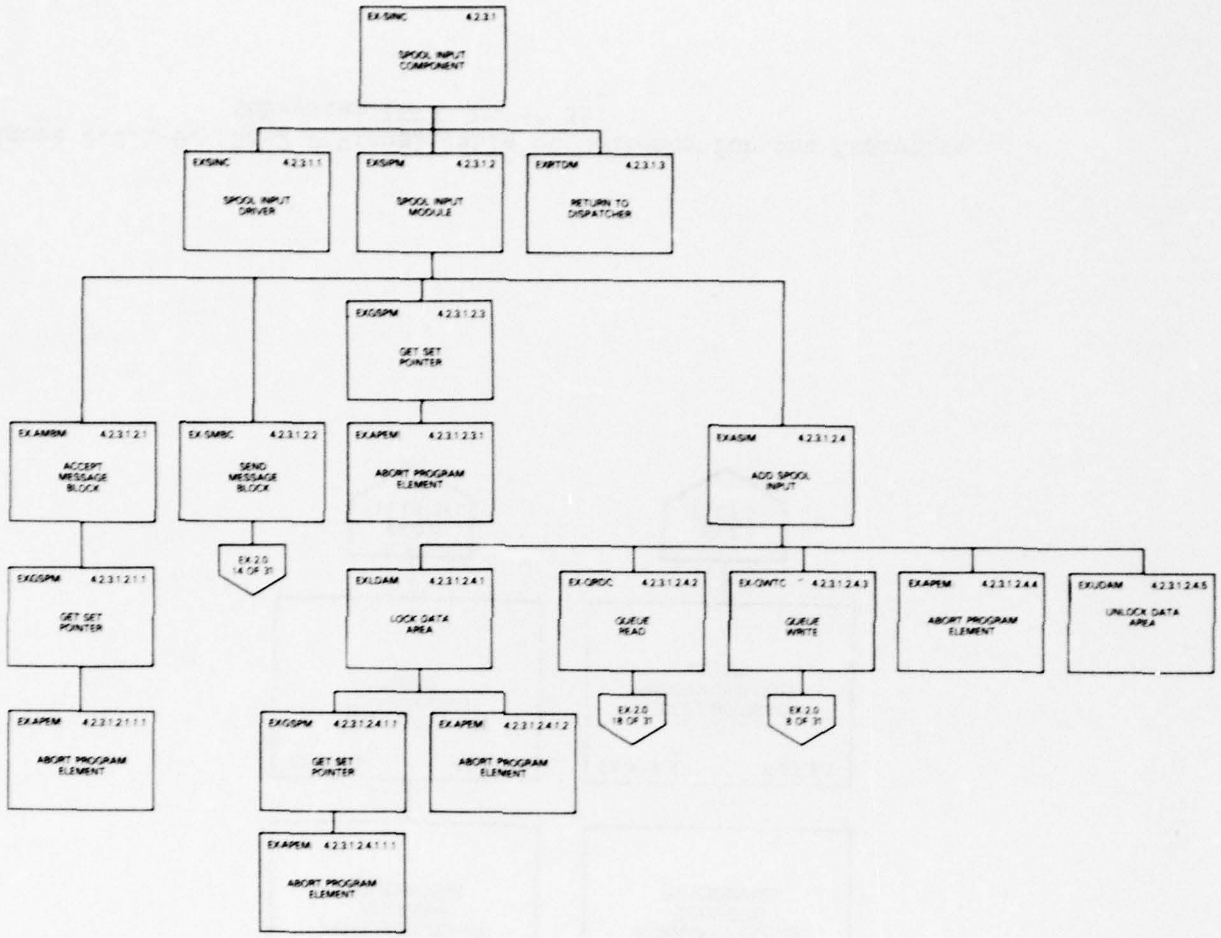


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (22 of 31)

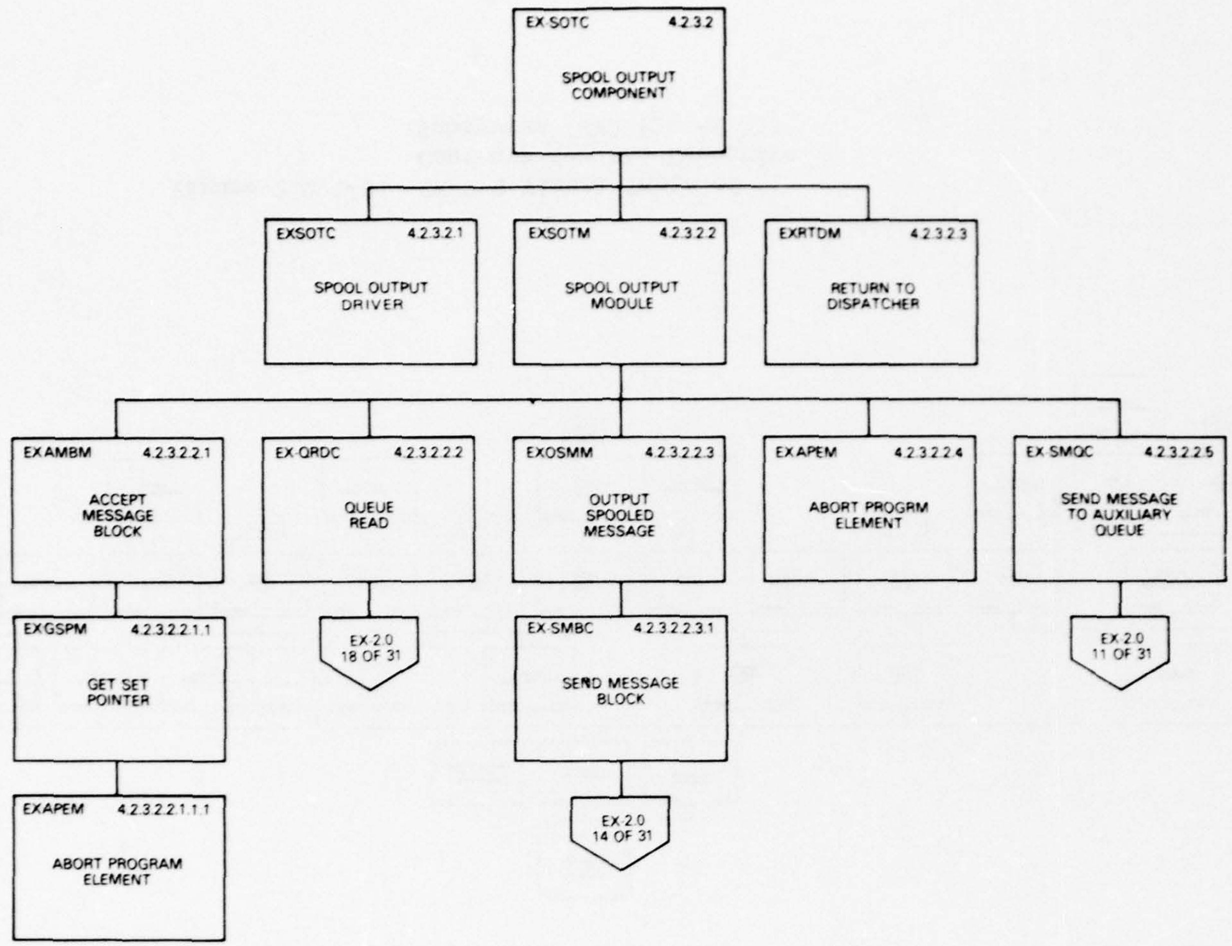


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (23 of 31)

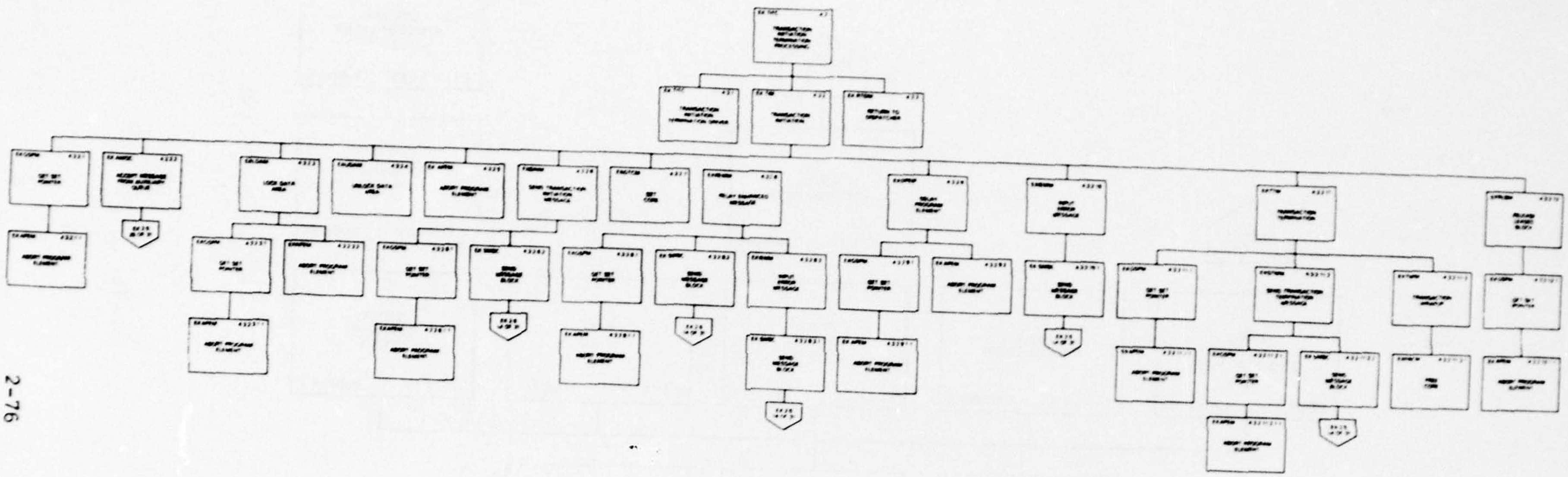


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (24 of 31)

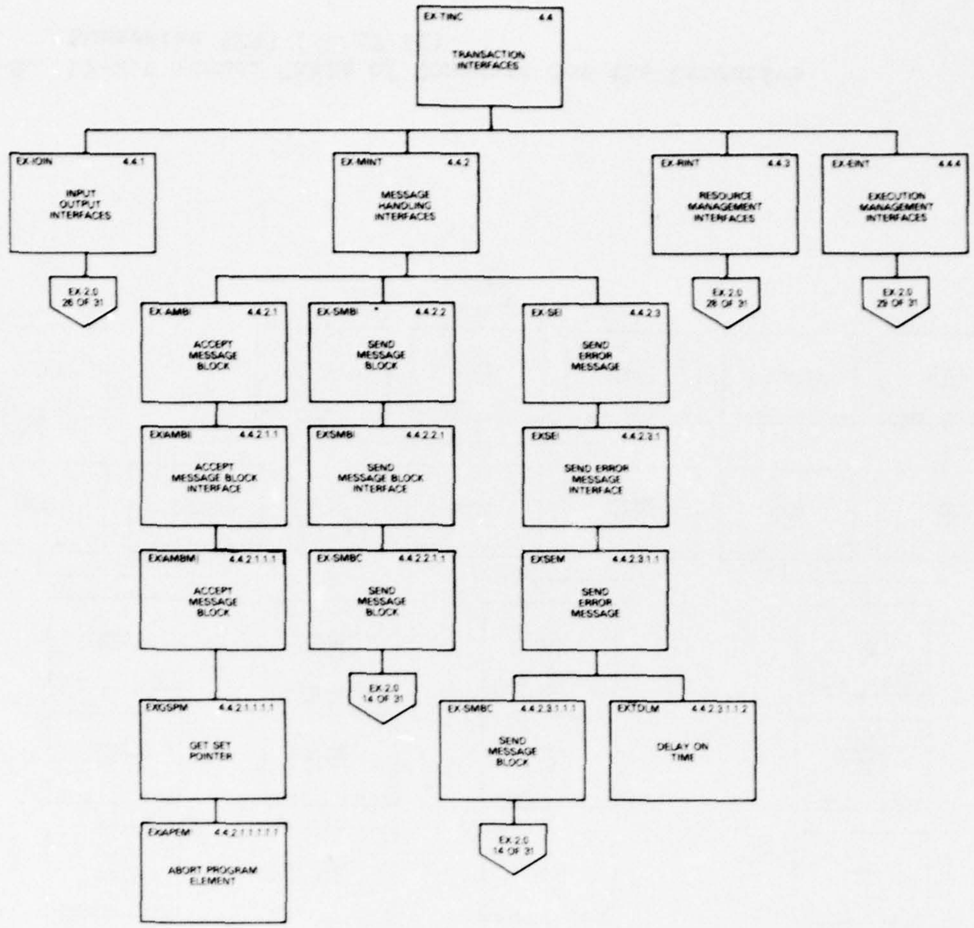


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (25 of 31)

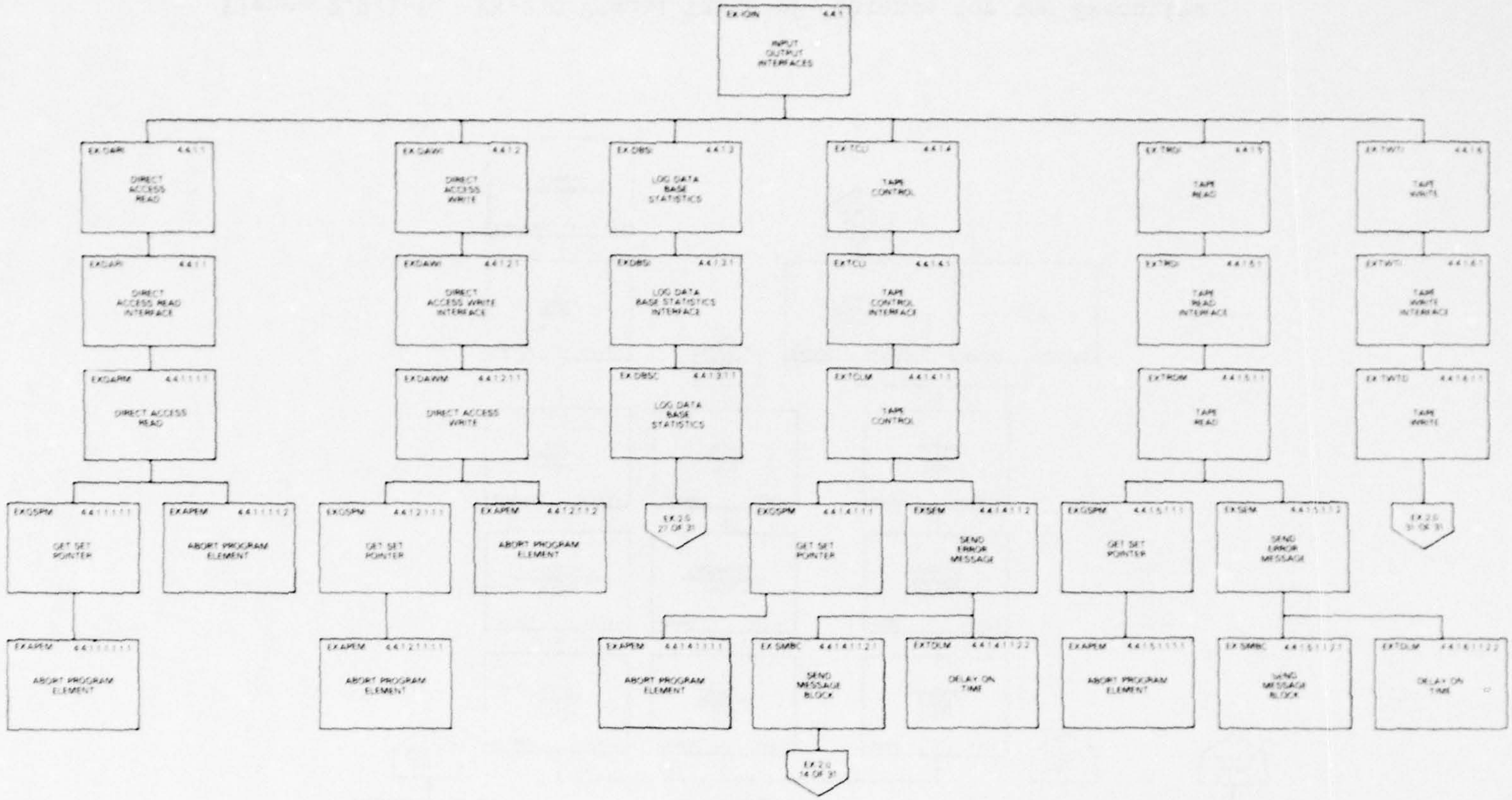


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (26 of 31)

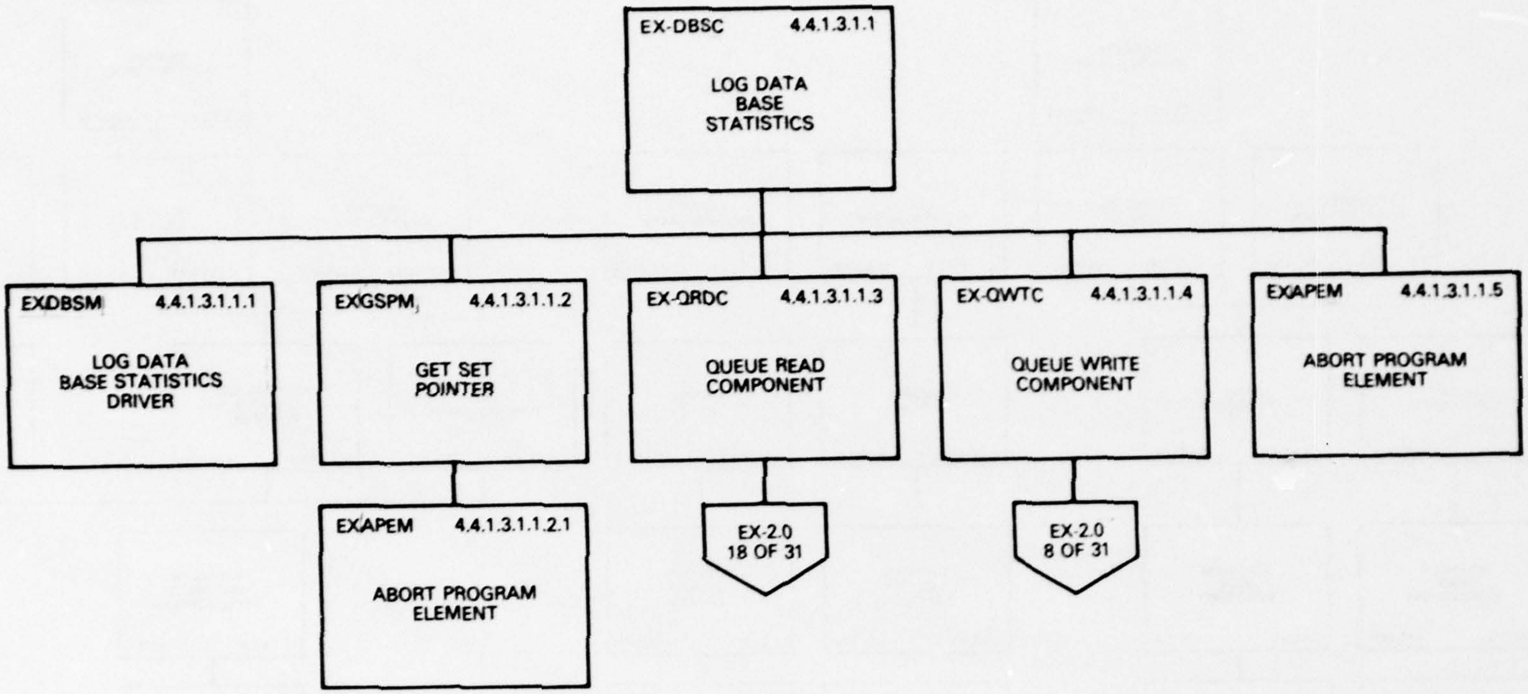


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (27 of 31)

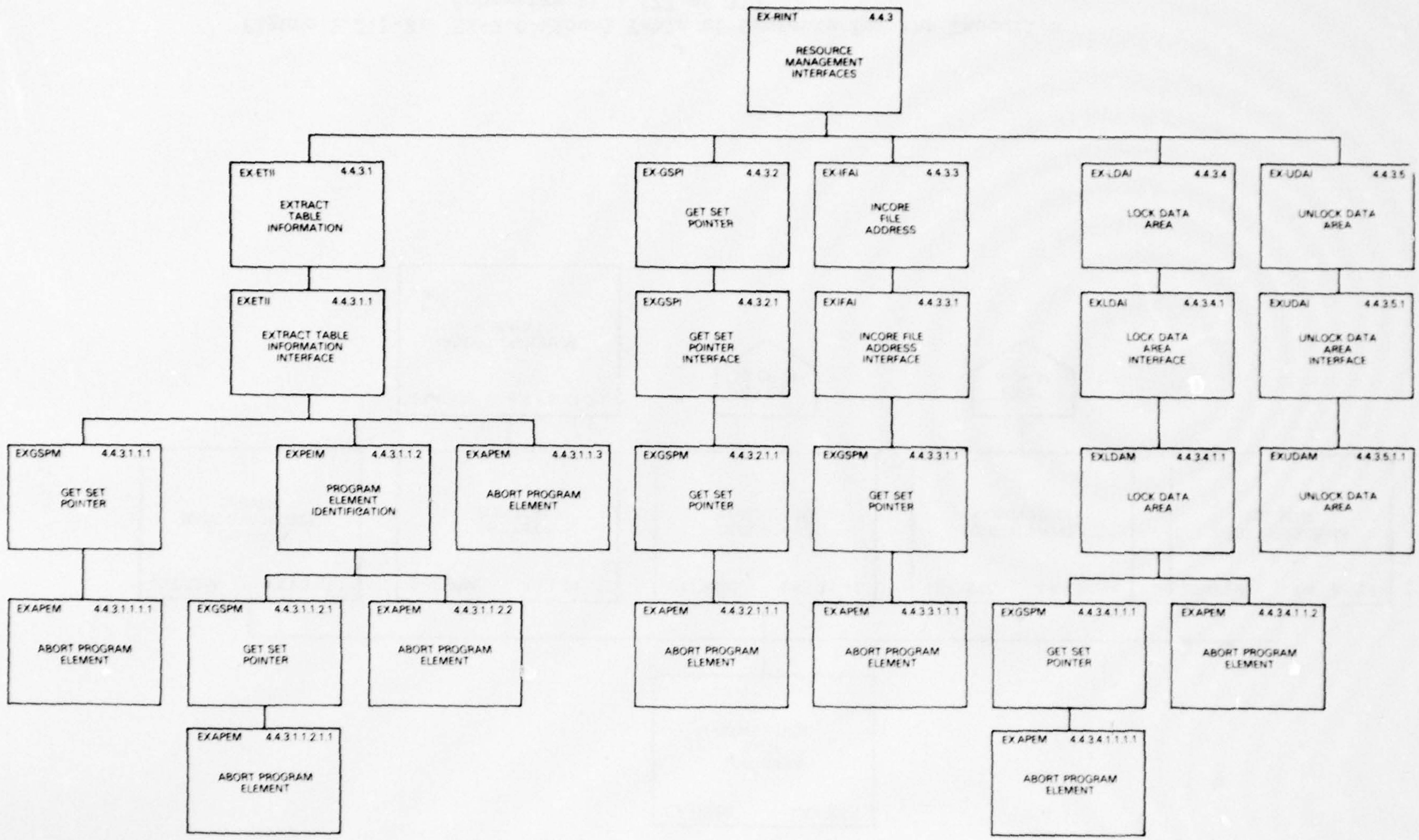


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (28 of 31)

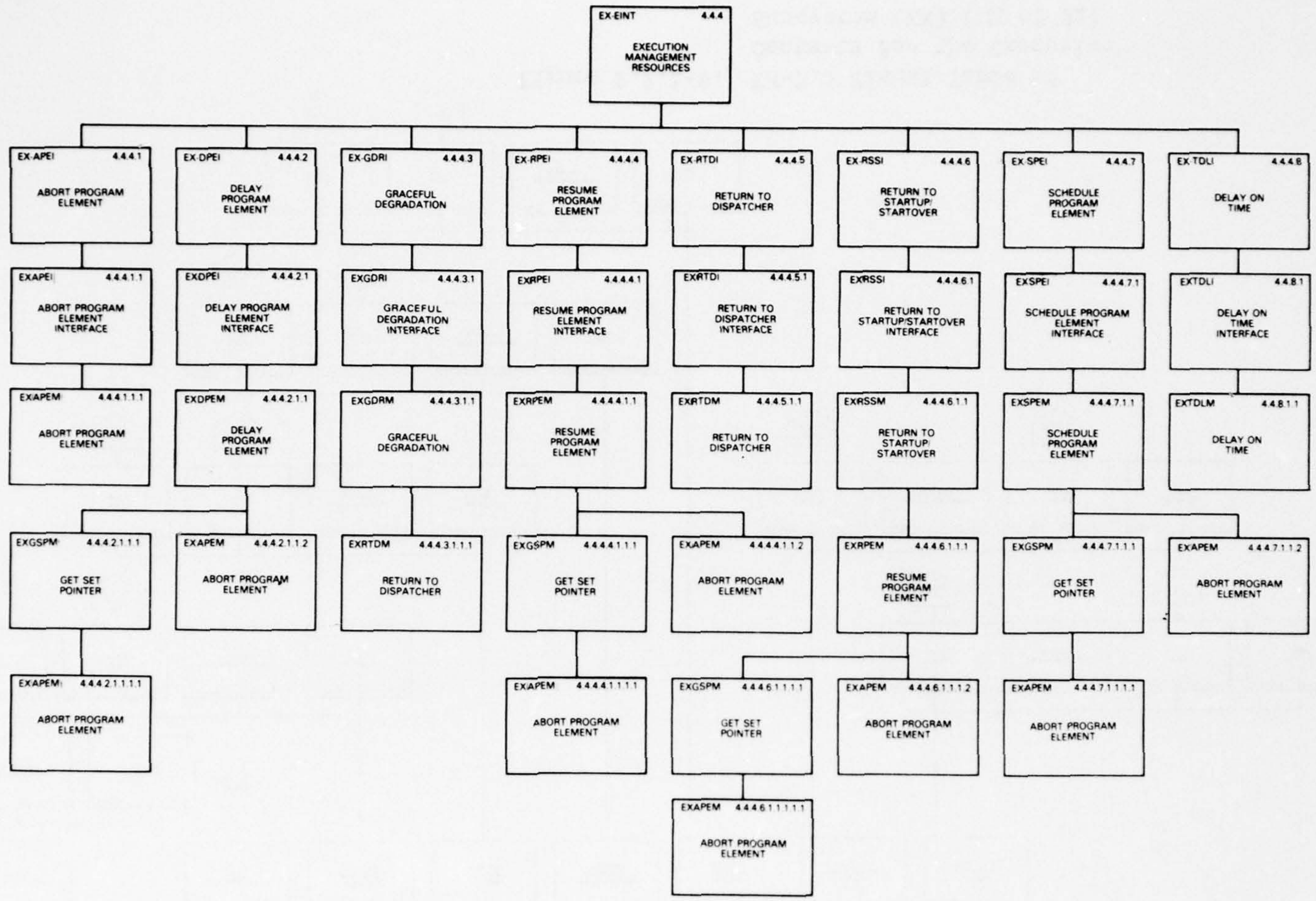


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (29 of 31)

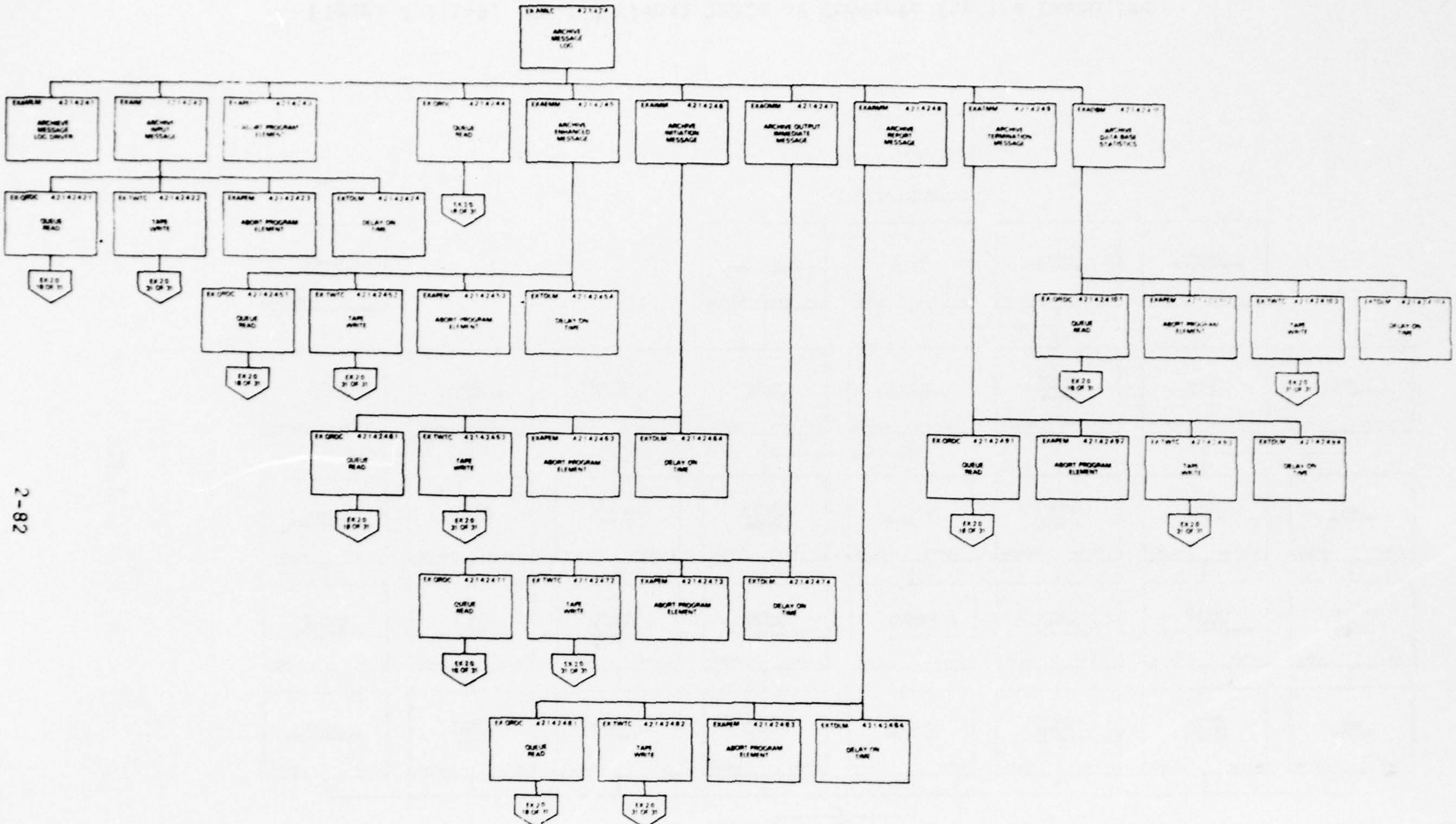


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (30 of 31)

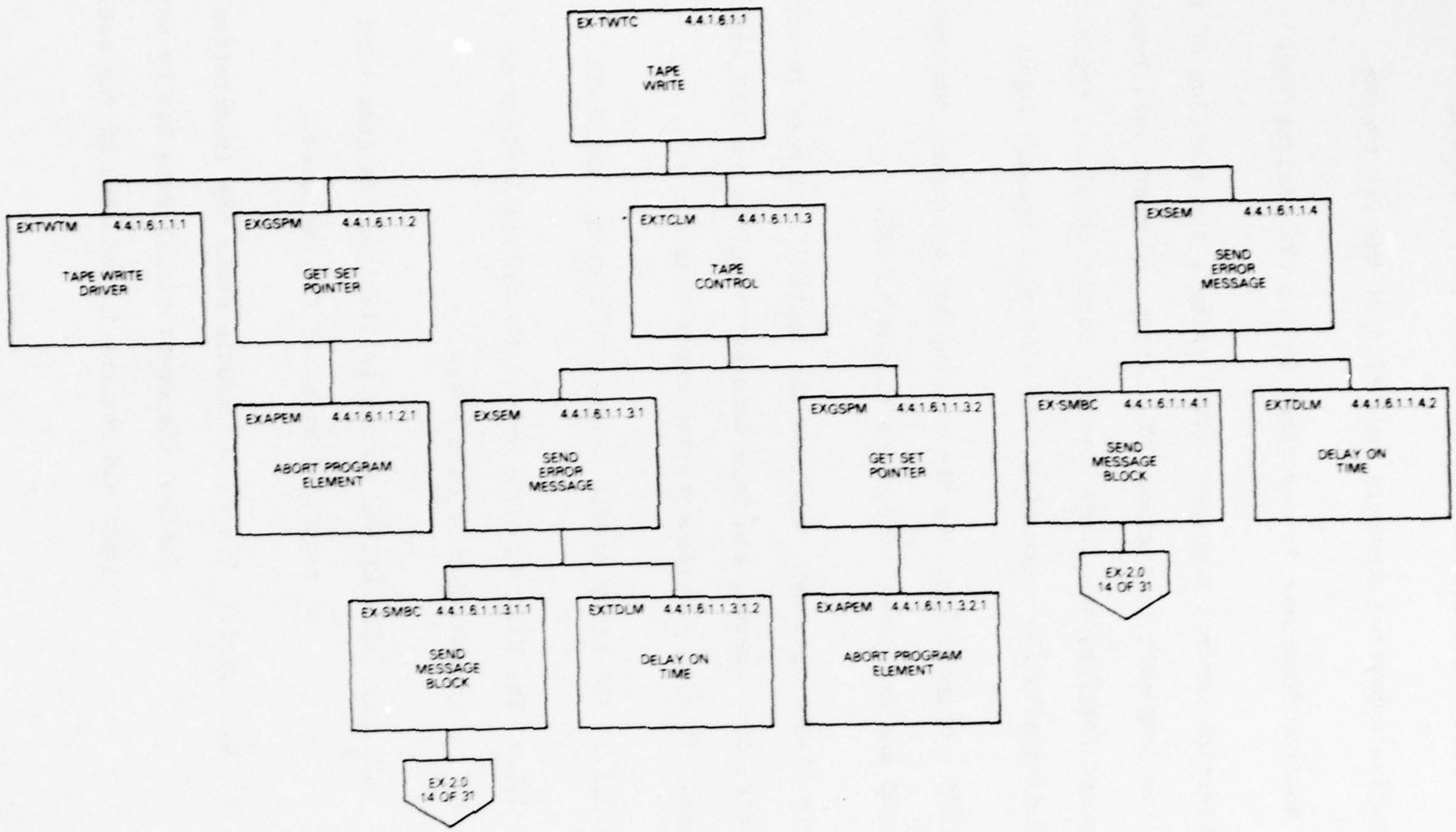


Figure 2.2.1-8. EX-2.0 Visual Table of Contents for the Executive Subsystem (EX) (31 of 31)

2.2.1.2.3 Description Section for the Executive (EX) Subsystem

2.0 The Executive Subsystem controls execution of the CFC system.

3.0 The CFC Monitor Component is described in the CFC Monitor SDD.

4.0 The Transaction Control Component (TCC) extends the function of the CFC Monitor Component to include CFC startup and startover, transaction message logging, auxiliary queuing, output spooling, transaction initiation/termination processing, and simulated message input.

4.1 The EX-TSSC component includes the startup and startover components. The startup and startover process is unique to CFC.

4.1.1 The EX-SUC component performs the initialization of in-core data, disk queues, and tape data set required for CFC startups. It also provides startup error statistics.

4.1.1.1 The EXSUM module drives CFC startup functions.

4.1.1.2 The EXIDAM module drives the initialization of CFC unique in-core data areas.

4.1.1.2.1 The EXIRCM module initializes the sets used for reconfiguration of CFC resources.

4.1.1.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.2.1.1.1 The EXAPEM module causes a program element to abort.

4.1.1.2.2 The EXIPEM module initializes the sets used for program element control.

4.1.1.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.2.2.1.1 The EXAPEM module causes a program element to abort.

4.1.1.2.3 The EXIQCM module initializes the sets used for queue management control.

4.1.1.2.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.2.3.1.1 The EXAPEM module causes a program element to abort.

4.1.1.2.4 The EXIFCM module initializes in-core files from disk.

4.1.1.2.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.2.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.2.5 The EXIGDM module initializes the global data sets.

4.1.1.2.5.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.2.5.1.1 The EXAPEM module causes a program element to abort.

4.1.1.2.5.2 The EXEPCM module computes the elapsed number of seconds from epoch time start to most recent midnight.

4.1.1.3 The EX-IDDC component initializes the Queue Management data sets.

4.1.1.3.1 The EXIDDM module is a driver module to initialize the Log Queue, Auxiliary Queue, and Spool Queue.

4.1.1.3.2 The EXDAOM module opens the direct access data sets.

4.1.1.3.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.2.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.2.2 The EXAPEM module causes a program element to abort.

4.1.1.3.3 The EXIAQM module initializes the Auxiliary Queue set by writing the data set descriptor entry and generating the free queue entry chain.

4.1.1.3.3.1 The EXSEM module sends an error message to the operator.

4.1.1.3.3.1.1 The EX-SMBC component sends a message block to another PE's queue. (See 4.1.2.4.5.7.)

4.1.1.3.3.1.2 The EXTDLM module delays a program element's execution.

4.1.1.3.3.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.3 The EXAPEM module causes a program element to abort.

4.1.1.3.3.4 The EXDAWM module performs a direct access write.

4.1.1.3.3.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.4.2 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5 The EX-QWTC component writes a block to a queue or deletes a block from a queue.

4.1.1.3.3.5.1 The EXQWTM module drives the queue write processing.

4.1.1.3.3.5.2 The EXLDAM module enqueues a data area.

4.1.1.3.3.5.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.2.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.2.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.3 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.3.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.4 The EXQFDM module finds a specific record in the queues by using a record pointer.

4.1.1.3.3.5.4.1 The EXDARM module performs a direct access read.

4.1.1.3.3.5.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.4.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.4.1.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.4.2 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.5 The EX-QDLC component deletes an entry from a queue.

4.1.1.3.3.5.5.1 The EXQDLM module is the queue delete driver.

4.1.1.3.3.5.5.2 The EXDARM module performs a direct access read.

4.1.1.3.3.5.5.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.5.2.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.5.2.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.5.3 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.5.4 The EX-QDLC component deletes an entry from a queue (see 4.1.1.3.3.5.5).

4.1.1.3.3.5.5.5 The EXDAWM module performs a direct access write.

4.1.1.3.3.5.5.5.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.5.5.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.5.5.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.6 The EXQADM module adds a directory entry to a queue's directory.

4.1.1.3.3.5.6.1 The EXDARM module performs a direct access read.

4.1.1.3.3.5.6.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.6.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.6.1.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.6.2 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.6.3 The EXDAWM module performs a direct access write.

4.1.1.3.3.5.6.3.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.6.3.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.6.3.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.7 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.8 The EXDAWM module performs a direct access write.

4.1.1.3.3.5.8.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.8.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.3.5.8.2.1 The EXAPEM module causes a program element to abort.

4.1.1.3.3.5.9 The EXUDAM module dequeues a data area.

4.1.1.3.4 The EXILQM module initializes the log queue directory.

4.1.1.3.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.4.2 The EXAPEM module causes a program element to abort.

4.1.1.3.4.3 The EXSEM module sends an error message to the operator.

4.1.1.3.4.3.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.1.1.3.4.3.2 The EXTDLM module delays a program element's execution.

4.1.1.3.4.4 The EXDAWM module performs a direct access write.

4.1.1.3.4.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.4.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.4.4.2 The EXAPEM module causes a program element to abort.

4.1.1.3.5 The EXISQM module initializes the Spool Queue set by writing the data set descriptor entry and generating the free queue entry chain.

4.1.1.3.5.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.5.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.5.2 The EXAPEM module causes a program element to abort.

4.1.1.3.5.3 The EXSEM module sends an error message to the operator.

4.1.1.3.5.3.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.1.1.3.5.3.2 The EXTDLM module delays a program element's execution.

4.1.1.3.5.4 The EXDAWM module performs a direct access write.

4.1.1.3.5.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.3.5.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.3.5.4.2 The EXAPEM module causes a program element to abort.

4.1.1.4 The EXITDM module controls the positioning and initialization of CFC unique tape data sets.

4.1.1.4.1 The EXIALM module rewinds the Archive Log Queue Tape to the load point.

4.1.1.4.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.4.1.1.1 The EXAPEM module causes a program element to abort.

4.1.1.4.1.2 The EXSEM module sends an error message to the operator.

4.1.1.4.1.2.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.1.1.4.1.2.2 The EXTDLM module delays a program element's execution.

4.1.1.4.1.3 The EXAPEM module causes a program element to abort.

4.1.1.4.1.4 The EXTCLM module performs control functions on CFC tapes.

4.1.1.4.1.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.1.4.1.4.1.1 The EXAPEM module causes a program element to abort.

4.1.1.4.1.4.2 The EXSEM module sends an error message to the operator.

4.1.1.4.1.4.2.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.1.1.4.1.4.2.2 The EXTDLM module delays a program element's execution.

4.1.1.5 The EXRTDM module returns control to the dispatcher.

4.1.2 The EX-SOC component performs the reinitialization and reconfiguration of CFC unique resources during startover including in-core data, disk queues, tape data sets and input messages.

4.1.2.1 The EXSOM module is a driver module to control all CFC unique startover processing.

4.1.2.2 The EXRDAM module reinitializes the in-core data area.

4.1.2.2.1 The EXIGDM module initializes the in-core global data areas.

4.1.2.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.2.1.1.1 The EXAPEM module causes a program element to abort.

4.1.2.2.1.2 The EXEPCM module computes the elapsed number of seconds from epoch time start to most recent midnight.

4.1.2.2.2 The EXIRCM module initializes the reconfiguration control sets.

4.1.2.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.2.2.1.1 The EXAPEM module causes a program element to abort.

4.1.2.2.3 The EXIPEM module initializes the PE tables.

4.1.2.2.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.2.3.1.1 The EXAPEM module causes a program element to abort.

4.1.2.2.4 The EXIFCM module initializes in-core files from disk.

4.1.2.2.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.2.4.1.1 The EXAPEM module causes a program element to abort.

4.1.2.2.5 The EXIQCM module initializes the sets used for queue management control.

4.1.2.2.5.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.2.5.1.1 The EXAPEM module causes a program element to abort.

4.1.2.3 The EXRRDM module initializes the directories for the auxiliary queue.

4.1.2.3.1 The EXIAQM initializes the auxiliary queue directory.

4.1.2.3.1.1 The EXSEM module sends an error message to the operator.

4.1.2.3.1.1.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.1.2.3.1.1.2 The EXAPEM module causes a program element to abort.

4.1.2.3.1.2 The EXAPEM module causes a program element to abort.

4.1.2.3.1.3 The EXDAWM module performs a direct access write.

4.1.2.3.1.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.3.1.3.1.1 The EXAPEM module causes a program element to abort.

4.1.2.3.1.3.2 The EXAPEM module causes a program element to abort.

4.1.2.3.1.4 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.3.1.4.1 The EXAPEM module causes a program element to abort.

4.1.2.3.1.5 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.1.2.3.2 The EXDAOM module opens direct access data sets.

4.1.2.3.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.3.2.1.1 The EXAPEM module causes a program element to abort.

4.1.2.3.2.2 The EXAPEM module causes a program element to abort.

4.1.2.4 The EX-IMSC component terminates or reinstates transactions.

4.1.2.4.1 The EXIMSM module controls the termination or reinstatement of transactions.

4.1.2.4.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.2.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3 The EX-QRDC component reads a block from a queue.

4.1.2.4.3.1 The EXQRDM module drives the queue read processing.

4.1.2.4.3.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.3.2.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.3 The EXLDAM module enqueues a data area.

4.1.2.4.3.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.3.3.1.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.3.2 The EXAPEM module causes a program element to abort.

4.1.2.4.3.4 The EXQFDM module finds a specified record in a queue by using a record pointer.

4.1.2.4.3.4.1 The EXDARM module performs a direct access read.

4.1.2.4.3.4.1.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.4.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.3.4.1.2.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.4.2 The EXAPEM module causes a program element to abort.

4.1.2.4.3.5 The EXDARM module performs a direct access read.

4.1.2.4.3.5.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.5.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.3.5.2.1 The EXAPEM module causes a program element to abort.

4.1.2.4.3.6 The EXAPEM module causes a program element to abort.

4.1.2.4.3.7 The EXUDAM module dequeues a data area.

4.1.2.4.4 The EXAPEM module causes a program element to abort.

4.1.2.4.5 The EXEIMM module builds an enhanced message from an input message.

4.1.2.4.5.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.5.1.1 The EXAPEM module causes a program element to abort.

4.1.2.4.5.2 The EXLDAM module enqueues a data area.

4.1.2.4.5.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.5.2.1.1 The EXAPEM module causes a program element to abort.

4.1.2.4.5.2.2 The EXAPEM module causes a program element to abort.

4.1.2.4.5.3 The EXUDAM module dequeues a data area.

4.1.2.4.5.4 The EXASDM module logs the spool ID for each input message.

4.1.2.4.5.4.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.1.2.4.5.4.2 The EXAPEM module causes a program element to abort.

4.1.2.4.5.5 The EXAPEM module causes a program element to abort.

4.1.2.4.5.6 The EXLEMM module controls the logging of enhanced messages.

4.1.2.4.5.6.1 The EXLEDM module logs the enhanced messages into the log queue.

4.1.2.4.5.6.1.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.1.2.4.5.6.1.2 The EXAPEM module causes a program element to abort.

4.1.2.4.5.7 The EX-SMBC component sends a message block to another PE's queue.

4.1.2.4.5.7.1 The EXSMBM module drives the send message processing.

4.1.2.4.5.7.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.5.7.2.1 The EXAPEM module causes a program element to abort.

4.1.2.4.5.7.3 The EXAPEM module causes a program element to abort.

4.1.2.4.5.7.4 The EXPEIM module returns the numeric and character identification of the PE in which it is called.

4.1.2.4.5.7.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.1.2.4.5.7.4.1.1 The EXAPEM module causes a program element to abort.

4.1.2.4.5.7.4.2 The EXAPEM module causes a program element to abort.

4.1.2.4.6 The EXTTPN module sends a termination message for a message that has an initiation but no termination message.

4.1.2.4.6.1 The EX-SMQC component sends a message block to a PE that supports auxiliary queuing (see 4.2.1.1.1.2.5.6).

4.1.2.4.6.2 The EXAPEM module causes a program element to abort.

4.1.2.4.7 The EXRTNM module starts transactions that were not initiated at the time of the start-over.

4.1.2.4.7.1 The EX-SMQC component sends a message block to a PE that supports auxiliary queuing (see 4.2.1.1.1.2.5.6).

4.1.2.5 The EXRTDM module returns control to the dispatcher.

4.2 The EX-TIOC component includes the transaction input and output processing, which is composed of the log, spool, and auxiliary queue processing.

4.2.1 The EX-LQC component includes the log queue processing as a transaction travels through the system.

4.2.1.1 The EX-ILQC component processes the input messages by logging and enhancing them.

4.2.1.1.1 The EX-NIMC component processes the enroute input messages.

4.2.1.1.1.1 The EXNIMC module controls the enroute input message processing.

4.2.1.1.1.2 The EXNIMM module processes enroute messages by logging them and enhancing them.

4.2.1.1.1.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.2 The EXAMBM module accepts a message block from the queue.

4.2.1.1.1.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.3 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.4 The EX-LIMC component logs input messages.

4.2.1.1.1.2.4.1 The EXLIM module controls the logging of input messages.

4.2.1.1.1.2.4.2 The EX-LDKC component logs input messages to disk.

4.2.1.1.1.2.4.2.1 The EXLDKM module controls the logging of input messages to disk.

4.2.1.1.1.2.4.2.2 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.1.2.4.2.3 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5 The EX-EIQC component builds an enhanced message from an input message for message sent to PE's supporting auxiliary queuing.

4.2.1.1.1.2.5.1 The EXEIQM module controls the construction of enhanced messages.

4.2.1.1.1.2.5.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.5.2.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.3 The EXLDAM module enqueues a data area.

4.2.1.1.1.2.5.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.5.3.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.3.2 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.4 The EXUDAM module dequeues a data area.

4.2.1.1.1.2.5.5 The EXASDM module logs the spool ID for each input message.

4.2.1.1.1.2.5.5.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.1.2.5.5.2 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6 The EX-SMQC component sends a message block to a PE that supports auxiliary queuing.

4.2.1.1.1.2.5.6.1 The EXSMQM module controls the sending of a message block to a PE that supports auxiliary queuing.

4.2.1.1.1.2.5.6.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.5.6.2.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6.3 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6.4 The EXLDAM module enqueues a data area.

4.2.1.1.1.2.5.6.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.5.6.4.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6.4.2 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6.5 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.1.2.5.6.6 The EXUDAM module dequeues a data area.

4.2.1.1.1.2.5.6.7 The EXPEIM module returns the numeric and character identification of the PE in which it is called.

4.2.1.1.1.2.5.6.7.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.1.2.5.6.7.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.6.7.2 The EXAPEM module causes a program element to abort.

4.2.1.1.1.2.5.7 The EXLEMM module controls the logging of enhanced messages.

4.2.1.1.1.2.5.7.1 The EXLEDM module logs the enhanced messages into the log queue.

4.2.1.1.1.2.5.7.1.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.1.2.5.7.1.2 The EXAPEM module causes a program element to abort.

4.2.1.1.1.3 The EXRTDM module returns control to the dispatcher.

4.2.1.1.2 The EX-SIMC component processes the SCC input messages.

4.2.1.1.2.1 The EXSIMC module controls the SCC input message processing.

4.2.1.1.2.2 The EXSIMM module processes the SCC input message by logging them and enhancing them.

4.2.1.1.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.2.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.2 The EXAMBM module accepts a message block.

4.2.1.1.2.2.2.1 The EXGSPM module causes a program element to abort.

4.2.1.1.2.2.3 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.4 The EX-LIMC component logs input messages.

4.2.1.1.2.2.4.1 The EXLIM module controls the logging of input messages.

4.2.1.1.2.2.4.2 The EX-LDKC component logs input messages to disk.

4.2.1.1.2.2.4.2.1 The EXLDKM module controls the logging of input messages to disk.

4.2.1.1.2.2.4.2.2 The EX-QWTC component writes a block to a queue or deletes a block from a queue.

4.2.1.1.2.2.4.2.3 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5 The EX-EIMC component builds an enhanced message from an input message.

4.2.1.1.2.2.5.1 The EXEIMM module controls the construction of enhanced messages.

4.2.1.1.2.2.5.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.2.2.5.2.1 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5.3 The EXLDAM module enqueues a data area.

4.2.1.1.2.2.5.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.2.2.5.3.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5.3.2 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5.4 The EXUDAM module dequeues a data area.

4.2.1.1.2.2.5.5 The EXASDM module logs the spool ID for each input message.

4.2.1.1.2.2.5.5.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.2.2.5.5.2 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5.6 The EXLEMM module controls the logging of enhanced messages.

4.2.1.1.2.2.5.6.1 The EXLEDM module logs the enhanced messages into the log queue.

4.2.1.1.2.2.5.6.1.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.1.2.2.5.6.1.2 The EXAPEM module causes a program element to abort.

4.2.1.1.2.2.5.7 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.2.1.1.2.2.5.8 The EXAPEM module causes a program element to abort.

4.2.1.1.2.3 The EXRTDM module returns control to the dispatcher.

4.2.1.1.3 The EX-CIMC component processes the CSI input messages.

4.2.1.1.3.1 The EXCIMC module obtains the work area for the component, activates EXCIMM, and releases work area when the component is completed.

4.2.1.1.3.2 The EXCIMM module drives the CSI input message processing.

4.2.1.1.3.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.3.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.3.2.2 The EXAMBM module accepts a message block.

4.2.1.1.3.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.1.3.2.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.1.3.2.3 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.2.1.1.3.3 The EXRTDM module returns control to the dispatcher.

4.2.1.2 The EX-IMLC component places the initiation message into the log queue.

4.2.1.2.1 The EXIMLC module obtains the work area for the component, activates EXIMLM and releases the work area when the component is completed.

4.2.1.2.2 The EXIMLM module drives the insertion of an initiation message in the log queue.

4.2.1.2.2.1 The EXAMBM module accepts a message block.

4.2.1.2.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.2.2.1.1.1 The EXAPEM module causes a program element to abort.

4.2.1.2.2.2 The EXLIMM module controls the logging of initiation messages.

4.2.1.2.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.2.2.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.2.2.2.2 The EXLIDM module logs an initiation message into the log queue.

4.2.1.2.2.2.2.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.2.2.2.2.2 The EXAPEM module causes a program element to abort.

4.2.1.2.3 The EXRTDM module returns control to the dispatcher.

4.2.1.3 The EX-OMLC component logs output immediate messages and sends them to the source device.

4.2.1.3.1 The EXOMLC module obtains the work area for the component, activates EXOMLM and releases the work area when the component is completed.

4.2.1.3.2 The EXOMLM module controls the logging and sending of output immediate messages.

4.2.1.3.2.1 The EXAMBM module accepts a message block.

4.2.1.3.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.3.2.1.1.1 The EXAPEM module causes a program element to abort.

4.2.1.3.2.2 The EXSOSM module sends an output immediate message to the CFC monitor for disposal to the message source.

4.2.1.3.2.2.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.2.1.3.2.2.2 The EXAPEM module causes a program element to abort.

4.2.1.3.2.3 The EXLOMM module controls the logging of output immediate messages.

4.2.1.3.2.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.3.2.3.1.1 The EXAPEM module causes a program element to abort.

4.2.1.3.2.3.2 The EXLODM module logs an output immediate message into the log queue.

4.2.1.3.2.3.2.1 The EXLDAM module enqueues a data area.

4.2.1.3.2.3.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.3.2.3.2.1.2 The EXGSPM module scans the Transaction Control Component

Global Data Set by set name and returns the address of the set.

4.2.1.3.2.3.2.1.2.1 The EXAPEM module causes a program element to abort.

4.2.1.3.2.3.2.2 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.3.2.3.2.3 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.3.2.3.2.4 The EXAPEM module causes a program element to abort.

4.2.1.3.2.3.2.5 The EXUDAM module dequeues a data area.

4.2.1.3.3 The EXRTDM module returns control to the dispatcher.

4.2.1.4 The EX-TMLC component logs the termination message to disk and backup tape. This component also deletes messages in the log queue that have already been processed.

4.2.1.4.1 The EXTMLC module obtains the work area for the component, activates EXTMLM, and releases the work area when the component is completed.

4.2.1.4.2 The EXTMLM module controls the logging of termination messages and the archiving of log queue message entries.

4.2.1.4.2.1 The EX-AMQC component accepts a message block where auxiliary queuing is supported.

4.2.1.4.2.1.1 The EXAMQM module controls the accept processing.

4.2.1.4.2.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.4.2.1.2.1 The EXAPEM module causes a program element to abort.

4.2.1.4.2.1.3 The EXPEIM module returns the numeric and character identification of the PE in which it is called.

4.2.1.4.2.1.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.4.2.1.3.1.1 The EXAPEM module causes a program element to abort.

4.2.1.4.2.1.4 The EXLDAM module enqueues a data area.

4.2.1.4.2.1.4.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.4.2.1.4.1.1 The EXAPEM module causes a program element to abort.

4.2.1.4.2.1.4.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.1.5 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.1.6 The EXAPEM module causes a program element to abort.

4.2.1.4.2.1.7 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.4.2.1.8 The EXUDAM module dequeues a data area.

4.2.1.4.2.2 The EXLTMM module controls logging the termination message to disk.

4.2.1.4.2.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.4.2.2.1.1 The EXAPEM module causes a program element to abort.

4.2.1.4.2.2.2 The EXLTDM module logs the termination message on to the disk.

4.2.1.4.2.2.2.1 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.4.2.2.2.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.3 The EXSLMM module searches the queue to determine if the last termination message has been processed.

4.2.1.4.2.3.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.3.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4 The EX-AMLC component creates the archive tape entries.

4.2.1.4.2.4.1 The EXAMLM module controls the creation of the archive tape entries.

4.2.1.4.2.4.2 The EXAIM module logs the input message on the archive log tape.

4.2.1.4.2.4.2.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.2.2 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.2.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.2.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.4 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.5 The EXAEMM module logs the enhanced message on the archive tape.

4.2.1.4.2.4.5.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.5.2 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.5.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.5.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.6 The EXAIMM module logs the initiation message to the archive log tape.

4.2.1.4.2.4.6.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.6.2 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.6.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.6.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.7 The EXAOMM module logs output immediate messages to the archive log tape.

4.2.1.4.2.4.7.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.7.2 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.7.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.7.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.8 The EXARMM module logs output report messages to the archive log tape.

4.2.1.4.2.4.8.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.8.2 The EX-TWTC component writes a block to tape (see 4.4.1.6.6.1).

4.2.1.4.2.4.8.3 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.8.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.9 The EXATMM module logs termination messages to the archive log tape.

4.2.1.4.2.4.9.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.9.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.9.3 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.9.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.4.10 The EXADBM module logs data base statistics messages to the archive log tape.

4.2.1.4.2.4.10.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.4.10.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.4.10.3 The EX-TWTC component writes a block to tape (see 4.4.1.6.1.1).

4.2.1.4.2.4.10.4 The EXTDLM module delays a program element's execution.

4.2.1.4.2.5 The EXDMLM module deletes input, enhanced, initiation, output, report and termination messages from the log and spool queues after the last termination message has been processed.

4.2.1.4.2.5.1 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.1.4.2.5.2 The EXAPEM module causes a program element to abort.

4.2.1.4.2.5.3 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.1.4.2.6 The EXRLBM module releases a leased block.

4.2.1.4.2.6.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.1.4.2.6.1.1 The EXAPEM module causes a program element to abort.

4.2.1.4.3 The EXRTDM module returns control to the dispatcher.

4.2.2 The EX-AQC component records and removes messages in the Auxiliary Queue.

4.2.2.1 The EX-AQEC component records messages into the Auxiliary Queue.

4.2.2.1.1 The EX-SMQC component sends a message block to a PE supporting auxiliary queuing (see 4.2.1.1.1.2.5.6).

4.2.2.2 The EX-AQDC component removes messages from the Auxiliary Queue.

4.2.2.2.1 The EX-AMQC component accepts a message block from a queue with Auxiliary Queuing Support (see 4.2.1.4.2.1).

4.2.3 The EX-SQC component includes all processing of the spool queue.

4.2.3.1 The EX-SINC component is responsible for the stacking of output from the Application Subsystem transaction load modules into the spool queue.

4.2.3.1.1 The EXSINC module obtains the work area for the component, activates EXSIPM, and releases the work area when the component is completed.

4.2.3.1.2 The EXSIPM module controls spooling of transaction output reports to the spool queue set.

4.2.3.1.2.1 The EXAMBM module accepts a message block from the queue.

4.2.3.1.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the addres: of the set.

4.2.3.1.2.1.1.1 The EXAPEM module causes a program element to abort.

4.2.3.1.2.2 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.2.3.1.2.3 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.3.1.2.3.1 The EXAPEM module causes a program element to abort.

4.2.3.1.2.4 The EXASIM module adds transaction output reports to the spool queue.

4.2.3.1.2.4.1 The EXLDAM module enqueues a data area.

4.2.3.1.2.4.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.3.1.2.4.1.1.1 The EXAPEM module causes a program element to abort.

4.2.3.1.2.4.1.2 The EXAPEM module causes a program element to abort.

4.2.3.1.2.4.2 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.3.1.2.4.3 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.2.3.1.2.4.4 The EXAPEM module causes a program element to abort.

4.2.3.1.2.4.5 The EXUDAM module dequeues a data area.

4.2.3.1.3 The EXRTDM module returns control to the dispatcher.

4.2.3.2 The EX-SOTC component is responsible for retrieving and deleting spool entries and spool IDs as they are processed.

4.2.3.2.1 The EXSOTC module obtains the work area for the component, activates EXSOTM, and releases the work area when the component is complete.

4.2.3.2.2 The EXSOTM module controls the sending of spooled output messages to the indicated output devices.

4.2.3.2.2.1 The EXAMBM module accepts a message from the queue.

4.2.3.2.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.2.3.2.2.1.1.1 The EXAPEM module causes a program element to abort.

4.2.3.2.2.2 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.2.3.2.2.3 The EXOSMM module sends an output report to selected output devices.

4.2.3.2.2.3.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.2.3.2.2.4 The EXAPEM module causes a program element to abort.

4.2.3.2.2.5 The EX-SMQC component sends a message block to a PE that supports auxiliary queuing (see 4.2.1.1.1.2.5.6).

4.2.3.2.3 The EXRTDM module returns control to the dispatcher.

4.3 The EX-TITC component performs the initiation and termination of Application Subsystem's load modules.

4.3.1 The EXTITC module obtains the work area for the component, activates EXTIM, and releases the work area when the component completed.

4.3.2 The EXTIM module controls the initiation of transaction load modules, puts itself in a wait state by issuing a CEASE SVC, and activates EXTTM for transaction termination processing.

4.3.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.1.1 The EXAPEM module causes a program element to abort.

4.3.2.2 The EX-AMQC component accepts a message block from the queue where auxiliary queuing is supported (see 4.2.1.4.2.1).

4.3.2.3 The EXLDAM module enqueues a data area.

4.3.2.3.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.3.1.1 The EXAPEM module causes a program element to abort.

4.3.2.4 The EXUDAM module dequeues a data area.

4.3.2.5 The EXAPEM module causes a program element to abort.

4.3.2.6 The EXSINM module constructs a transaction initiation message and sends it to the initiation message log component.

4.3.2.6.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.6.1.1 The EXAPEM module causes a program element to abort.

4.3.2.6.2 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.3.2.7 The EXGTCM module acquires main storage.

4.3.2.8 The EXREMM module relays the enhanced message to the transaction PE.

4.3.2.8.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.8.1.1 The EXAPEM module causes a program element to abort.

4.3.2.8.2 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.3.2.8.3 The EXIEMM module constructs an error message and sends it to EXOMLC to be sent to the user.

4.3.2.8.3.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.3.2.9 The EXDPEM module delays a PE's execution.

4.3.2.9.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.9.1.1 The EXAPEM module causes a program element to abort.

4.3.2.9.2 The EXAPEM module causes a program element to abort.

4.3.2.10 The EXIEMM module constructs an error message and sends it to EXOMLC to be sent to the user.

4.3.2.10.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.3.2.11 The EXTTM module processes the termination of a message.

4.3.2.11.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.11.1.1 The EXAPEM module causes a program element to abort.

4.3.2.11.2 The EXSTMM module constructs a transaction termination message and sends it to the termination message log component.

4.3.2.11.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.11.2.1.1 The EXAPEM module causes a program element to abort.

4.3.2.11.2.2 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.3.2.11.3 The EXTWM module performs wrapup functions such as releasing the transaction initiation/ termination control block.

4.3.2.11.3.1 The EXFRCM module frees main storage.

4.3.2.12 The EXRLBM module releases a leased block.

4.3.2.12.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.3.2.12.1.1 The EXAPEM module causes a program element to abort.

4.3.3 The EXRTDM module returns control to the dispatcher.

4.4 The EX-TINC component includes the monitor interfaces supplied by the Transaction Control Component for the Application Subsystem.

4.4.1 The EX-IOIN component includes those interfaces providing input and output services.

4.4.1.1 The EX-DARI component performs a direct access read.

4.4.1.1.1 The EXDARI module interfaces the JOVIAL calling standards with EXDARM.

4.4.1.1.1.1 The EXDARM module reads a block from a file in core or on disk.

4.4.1.1.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.1.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.1.1.1.1.2 The EXAPEM module causes a program element to abort.

4.4.1.2 The EX-DAWI component performs a direct access write.

4.4.1.2.1 The EXDAWI module interfaces the JOVIAL calling standards with EXDAWM.

4.4.1.2.1.1 The EXDAWM module writes a block to a file in core or on disk.

4.4.1.2.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.2.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.1.2.1.1.2 The EXAPEM module causes a program element to abort.

4.4.1.3 The EX-DBSI component logs a data base statistic to disk.

4.4.1.3.1 The EXDBSI module interfaces the JOVIAL calling standards with EXDBSM.

4.4.1.3.1.1 The EX-DBSC component logs a data base statistic into the log queue set.

4.4.1.3.1.1.1 The EXDBSM module controls the logging of data base statistics.

4.4.1.3.1.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.3.1.1.2.1 The EXAPEM module causes a program element to abort.

4.4.1.3.1.1.3 The EX-QRDC component reads a block from a queue (see 4.1.2.4.3).

4.4.1.3.1.1.4 The EX-QWTC component writes a block to a queue or deletes a block from a queue (see 4.1.1.3.3.5).

4.4.1.3.1.1.5 The EXAPEM module causes a program element to abort.

4.4.1.4 The EX-TCLI component performs control functions on tapes.

4.4.1.4.1 The EXTCLI module interfaces the JOVIAL calling standards with EXTCLM.

4.4.1.4.1.1 The EXTCLM module controls the tape control functions.

4.4.1.4.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.4.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.1.4.1.1.2 The EXSEM module sends an error message to the operator.

4.4.1.4.1.1.2.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.1.4.1.1.2.2 The EXTDLM module delays a program element's execution.

4.4.1.5 The EX-TRDI component reads a block from tape.

4.4.1.5.1 The EXTRDI module interfaces the JOVIAL calling standards with EXTRDM.

4.4.1.5.1.1 The EXTRDM module reads a block of data from tape into core.

4.4.1.5.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.5.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.1.5.1.1.2 The EXSEM module sends an error message to the operator.

4.4.1.5.1.1.2.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.1.5.1.1.2.2 The EXTDLM module delays a program element's execution.

4.4.1.6 The EX-TWTI component writes a block to tape.

4.4.1.6.1 The EXTWTI module interfaces the JOVIAL calling standards with EXTWTM.

4.4.1.6.1.1 The EX-TWTC component writes a block to tape.

4.4.1.6.1.1.1 The EXTWTM module controls the tape write processing.

4.4.1.6.1.1.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.6.1.1.2.1 The EXAPEM module causes a program element to abort.

4.4.1.6.1.1.3 The EXTCLM module controls the tape control functions.

4.4.1.6.1.1.3.1 The EXSEM module sends an error message to the operator.

4.4.1.6.1.1.3.1.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.1.6.1.1.3.1.2 The EXTDLM module delays a program element's execution.

4.4.1.6.1.1.3.2 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.1.6.1.1.3.2.1 The EXAPEM module causes a program element to abort.

4.4.1.6.1.1.4 The EXSEM module sends an error message to the operator.

4.4.1.6.1.1.4.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.1.6.1.1.4.2 The EXTDLM module delays a program element's execution.

4.4.2 The EX-MINT component includes those interfaces providing message handling services.

4.4.2.1 The EX-AMBI component accepts a message block from the queue.

4.4.2.1.1 The EXAMBI module interfaces the JOVIAL calling standard with EXAMBM.

4.4.2.1.1.1 The EXAMBM module accepts a message block from the queue.

4.4.2.1.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.2.1.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.2.2 The EX-SMBI component sends a message block to another PE's queue.

4.4.2.2.1 The EXSMBI module interfaces the JOVIAL calling standard with EXSMBM.

4.4.2.2.1.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.2.3 The EX-SEI component sends an error message to the operator.

4.4.2.3.1 The EXSEI module interfaces the JOVIAL calling standard with EXSEM.

4.4.2.3.1.1 The EXSEM module sends an error message to the operator.

4.4.2.3.1.1.1 The EX-SMBC component sends a message block to another PE's queue (see 4.1.2.4.5.7).

4.4.2.3.1.1.2 The EXTDLM module delays a program element's execution.

4.4.3 The EX-RINT component includes the interfaces providing resource management services.

4.4.3.1 The EX-ETII component returns data base work area address and size, data base update security mask, and the message ID for a transaction.

4.4.3.1.1 The EXETII module controls the extraction of table information.

4.4.3.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.3.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.3.1.1.2 The EXPEIM module returns the numeric and character identification of the PE in which it is called.

4.4.3.1.1.2.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.3.1.1.2.1.1 The EXAPEM module causes a program element to abort.

4.4.3.1.1.2.2 The EXAPEM module causes a program element to abort.

4.4.3.1.1.3 The EXAPEM module causes a program element to abort.

4.4.3.2 The EX-GSPI component returns the address of a set.

4.4.3.2.1 The EXGSPI module interfaces the JOVIAL calling standard with EXGSPM.

4.4.3.2.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.3.2.1.1.1 The EXAPEM module causes a program element to abort.

4.4.3.3 The EX-IFAI component locates the in-core address of core-resident data sets.

4.4.3.3.1 The EXIFAI module controls the location of the address of a core-resident data set.

4.4.3.3.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.3.3.1.1.1 The EXAPEM module causes a program element to abort.

4.4.3.4 The EX-LDAI component enqueues a data area.

4.4.3.4.1 The EXLDAI module interfaces the JOVIAL calling standard with EXLDAM.

4.4.3.4.1.1 The EXLDAM module enqueues a data area.

4.4.3.4.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.3.4.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.3.4.1.1.2 The EXAPEM module causes a program element to abort.

4.4.3.5 The EX-UDAI component dequeues a data area.

4.4.3.5.1 The EXUDAI module interfaces the JOVIAL calling standard with EXUDAM.

4.4.3.5.1.1 The EXUDAM module dequeues a data area.

4.4.4 The EX-EINT component includes those interfaces providing execution management services.

4.4.4.1 The EX-APEI component aborts the current program element.

4.4.4.1.1 The EXAPEI module interfaces the JOVIAL calling standard with EXAPEM.

4.4.4.1.1.1 The EXAPEM module causes a program element to abort.

4.4.4.2 The EX-DPEI component delays the current program element.

4.4.4.2.1 The EXDPEI module interfaces the JOVIAL calling standard with EXDPEM.

4.4.4.2.1.1 The EXDPEM module delays the execution of the current PE until a target PE finishes or optionally until the PE is RESUME'd by another PE.

4.4.4.2.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.4.2.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.4.2.1.1.2 The EXAPEM module causes a program element to abort.

4.4.4.3 The EX-GDRI component discontinues processing of a program element.

4.4.4.3.1 The EXGDRI module interfaces the JOVIAL calling standard with EXGDRM.

4.4.4.3.1.1 The EXGDRM module discontinues processing a PE.

4.4.4.3.1.1.1 The EXRTDM module returns control to the dispatcher.

4.4.4.4 The EX-RPEI component resumes the execution of a delayed program element.

4.4.4.4.1 The EXRPEI module interfaces the JOVIAL calling standard with EXRPEM.

4.4.4.4.1.1 The EXRPEM module resumes the execution of a CEASE'd PE.

4.4.4.4.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.4.4.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.4.4.1.1.2 The EXAPEM module causes a program element to abort.

4.4.4.5 The EX-RTDI component returns control to the dispatcher.

4.4.4.5.1 The EXRTDI module interfaces the JOVIAL calling standard with EXRTDM.

4.4.4.5.1.1 The EXRTDM module returns control to the dispatcher.

4.4.4.6 The EX-RSSI component returns control to startup or startover.

4.4.4.6.1 The EXRSSI module interfaces the JOVIAL calling standard with EXRSSM.

4.4.4.6.1.1 The EXRSSM module provides EXRPEM with the IQU PE name.

4.4.4.6.1.1.1 The EXRPEM module resumes the execution of a CEASE'd PE.

4.4.4.6.1.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.4.6.1.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.4.6.1.1.1.2 The EXAPEM module causes a program element to abort.

4.4.4.7 The EX-SPEI component schedules a PE for execution.

4.4.4.7.1 The EXSPEI module interfaces the JOVIAL calling standard with EXSPEM.

4.4.4.7.1.1 The EXSPEM module schedules a PE for later execution.

4.4.4.7.1.1.1 The EXGSPM module scans the Transaction Control Component Global Data Set by set name and returns the address of the set.

4.4.4.7.1.1.1.1 The EXAPEM module causes a program element to abort.

4.4.4.7.1.1.2 The EXAPEM module causes a program element to abort.

4.4.4.8 The EX-TDLI component delays processing.

4.4.4.8.1 The EXTDLI module interfaces the JOVIAL calling standard with EXTDLM.

4.4.4.8.1.1 The EXTDLM module delays a program element's execution.

2.2.1.3 Data Base Management (DB) Subsystem

a. Purpose. The purpose of the Data Base Management (DB) Subsystem is to provide the mechanism for performance of all data base creation, inspection, update, validation, storage, retrieval, and security-checking functions within the CFC Software System. This subsystem supports the data base access needs of the OPCX, and,in addition, provides data base access functions for some of the major components of the SPCX, including DA, RA, and TR.

b. Subsystem Overview. The Data Base Management Subsystem consists primarily of logical file handler primitives which are invoked by OPCX modules and SPCX modules for the purpose of accessing or updating the data base. While providing these data base accesses and update capabilities, this subsystem must maintain data base integrity by performing data validity checking during data base access and update, and by performing data base update security checks. Data base startup and recovery procedures are provided with the capability to create new data base segments or to delete those segments which are no longer needed by the System.

Additionally, the Data Base Management Subsystem is responsible for a set of interface components which are used to coordinate the translation of message processor data requirements into a meaningful set of calls on Data Base Management Subsystem primitives. These components filter flight records based on a set of given qualifiers, extract information from individual flight records, and place this information in output arrays to be passed back to the calling modules.
Figure 2.2.1-9 presents the Data Base Management Subsystem Overview.

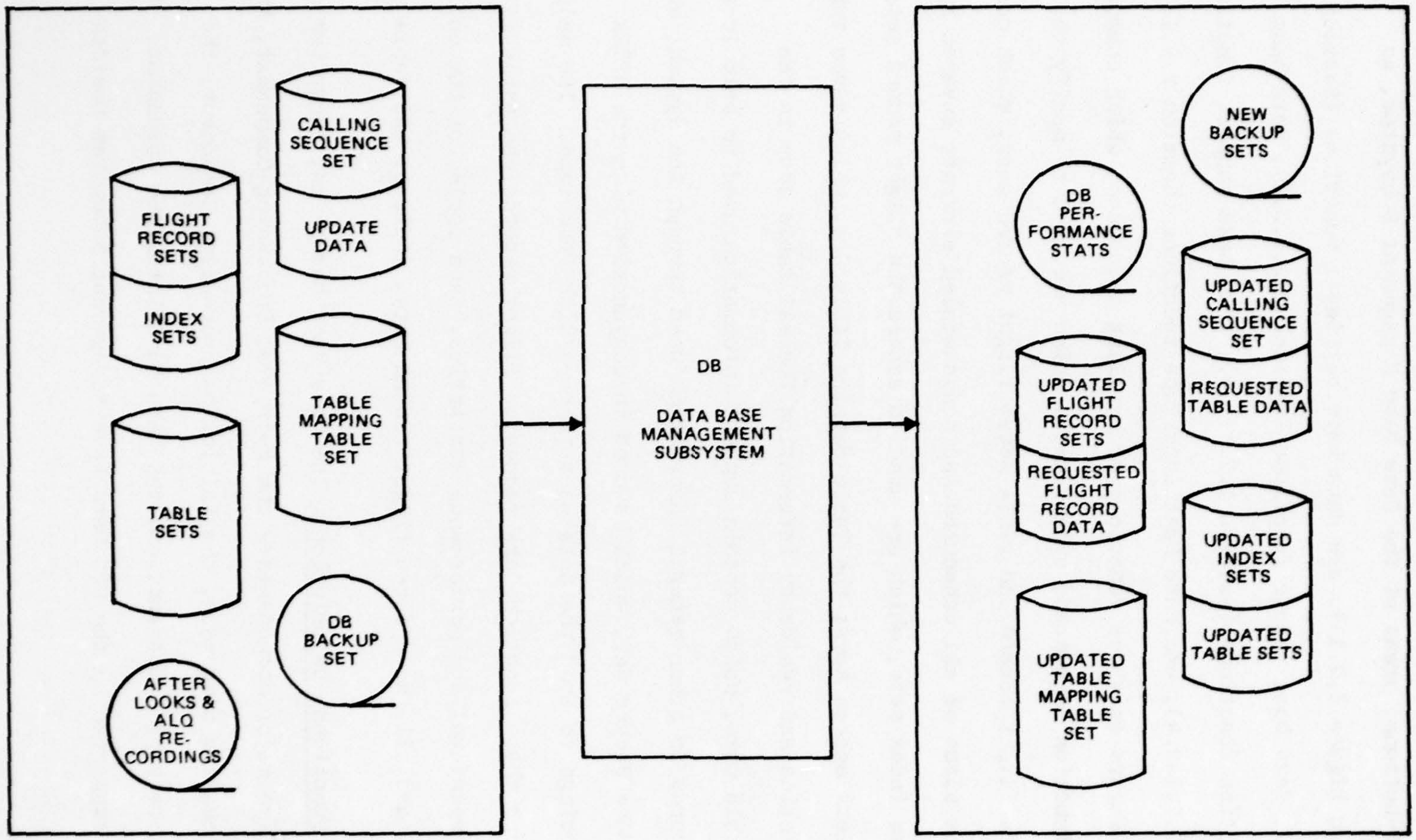


Figure 2.2.1-9. DB-1.0 Data Base Management Subsystem (DB) Overview

The five functional areas of the Data Base Management Subsystem, as depicted in Figure 2.2.1-9, are data base retrieval functions (Section 2.2.1.3.1), data base update functions (Section 2.2.1.3.2), data base initialization functions (Section 2.2.1.3.3), data base support functions (Section 2.2.1.3.4), and data base interface functions (Section 2.2.1.3.5). The major inputs to these functions are calling sequences, which communicate the function requested; update data, which are used to modify the appropriate flight record and table sets; flight record sets, which contain flight plans of all scheduled and nonscheduled aircraft covered by the system; index sets, which are used to access the flight record sets using various access keys; the Table Mapping Table Set, which gives table characteristics and residency information for all table sets in the System; table sets, which contain support information used to help locate flight records or other general information used through the System; and the Data Base Backup Set, which is used in conjunction with the AFTER LOOK recordings to load the data base to core/disk from tape. The major outputs from these functions are requested flight records and table data, data base retrieval and performance statistics, tape copies of the online data base, and all the updated flight record sets, indices, and table sets.

c. Functional Description. The Data Base Management Subsystem consists of five major components: the Retrieval Processing Component, the Update Processing Component, the Initialize Processing Component, the Support Processing Component, and the Data Base Interface Component. The data base supported by the CFC Data Base Management Subsystem resides

primarily on disk while some table and index sets reside in core. The two major sets within the data base are one static (OAG) and one dynamic (Non-OAG) flight record set supported by three index sets. Numerous informational table sets reside in the data base for use by the message processor and offline support components.

The Data Base Management Subsystem (DB) functions as the interface between all non-DB modules and the data base. For a description of the CFC data base, see Section 3. In the CFC System, no non-DB module has direct access to the data base. This requirement is absolutely essential to the proper functioning of the CFC System. The validity of the request for data access is checked by the DB Subsystem and all data to be written to the data base is validated before any write functions are performed. Before update data is written to the data base, a security check is performed by the DB Subsystem. Only authorized OPCX modules are allowed to update the data base online. Data base access and performance statistics are also maintained by the DB Subsystem for later use in producing performance measurement reports. Data base integrity maintenance and the collection of data base performance measurement statistics would be completely circumvented if an application or offline support component directly accessed the data base. That is the reason this practice is not allowed.

The Data Base Management Subsystem provides the following services for the calling modules:

DB-Service	Function
DB-CTAB	Create Table. Reserve disk space for the specified table, initialize all blocks to zeros, initialize the table header, and make a Table Mapping Table entry

DB-Service	Function
DB-CTBE	Create Table Entry. Insert key and associated entry into a table. Adjust pointers within table as required
DB-TABT	Get Table Entry. If a key is specified, retrieve the table entry associated with the key; if entry number is specified, retrieve the entry at that relative position in the table
DB-SETT	Set Table Entry. Replace the current table entry for the specified key with a new entry
DB-DTAB	Delete Table. Mark the Table Mapping Table entry for the specified table nonexistent
DB-CFRS	Create Flight Record Set. Initialize the OAG data set as a flight record set. Initialize the OAG Flight Index Set, Flight Accession Table, and Arrival/Departure Table
DB-INST	Insert Flight Record. Insert a new flight record into a flight record set. Update the Flight Index Set, Flight Accession Table, and Arrival/Departure Table, if required
DB-GETR	Get Flight Record. Retrieve a flight record from a flight record set. The flight record may be identified by aircraft ID, and/or by air carrier designation, and/or by arrival terminal or center, and/or by departure terminal or center, or by record number within the flight record set
DB-GETE	Get Next Flight Record. Retrieve the next flight record that satisfies the identification criteria established by the last DB-GETR
DB-CHGR	Change Flight Record. Replace the flight record at the specified location in the flight record set with a new flight record
DB-GTFR	Get Flight Record Interface. This is the interface between the application message processors (modules in the Application and Simulation Subsystems) and the DB-GETR and DB-GETE primitives. Filter out flight records returned from the DB primitives that do not meet specific selection criteria, extract specified data items from qualifying records and place in output arrays

DB-Service	Function
DB-UPFR	Update Flight Record Interface. This is the interface between the application message processor components and the DB-CHGR and DB-INST primitives. Based on information contained in the input arrays, read the flight record to be changed, update the record and have it written back to the data base

Note that the application interface components DB-GTFR and DB-UPFR process flight information on a record basis. Data are retrieved from the data base on a block basis, but records are passed to the interface routine one at a time. For the update processor, updated records are passed to the DB primitives one record at a time; they are then written to the data base on a block basis.

In a flight record, if the status indicator is two, the record is considered by the system as having been removed. This indicator can be set to two as part of the Change Record (CHGR) function.

The DB Subsystem will support the following functions, which are activated via an operator key-in or the EX Subsystem:

DB-Service	Function
DB-STUP	Startup. Optionally create an online data base from (1) a tape backup data base and (2) subsequent transactions performed since the tape backup was created on a disk version of the data base
DB-BKUP	Backup. Make a complete copy of the data base while updates are prevented

d. Major Components. The DB Subsystem is logically divided into four separate components.

These are:

- Retrieval Processing Component. This component performs all the functions necessary to access the data base and return all requested data to the calling module.

- Update Processing Component. This component performs all the functions necessary to transfer the data provided by the calling module to its proper location in the data base.

- Initialize Processing Component. This component performs all the functions necessary to initialize the flight record sets and their associated indices and to enter a new table set into the data base or to remove a table set from it.

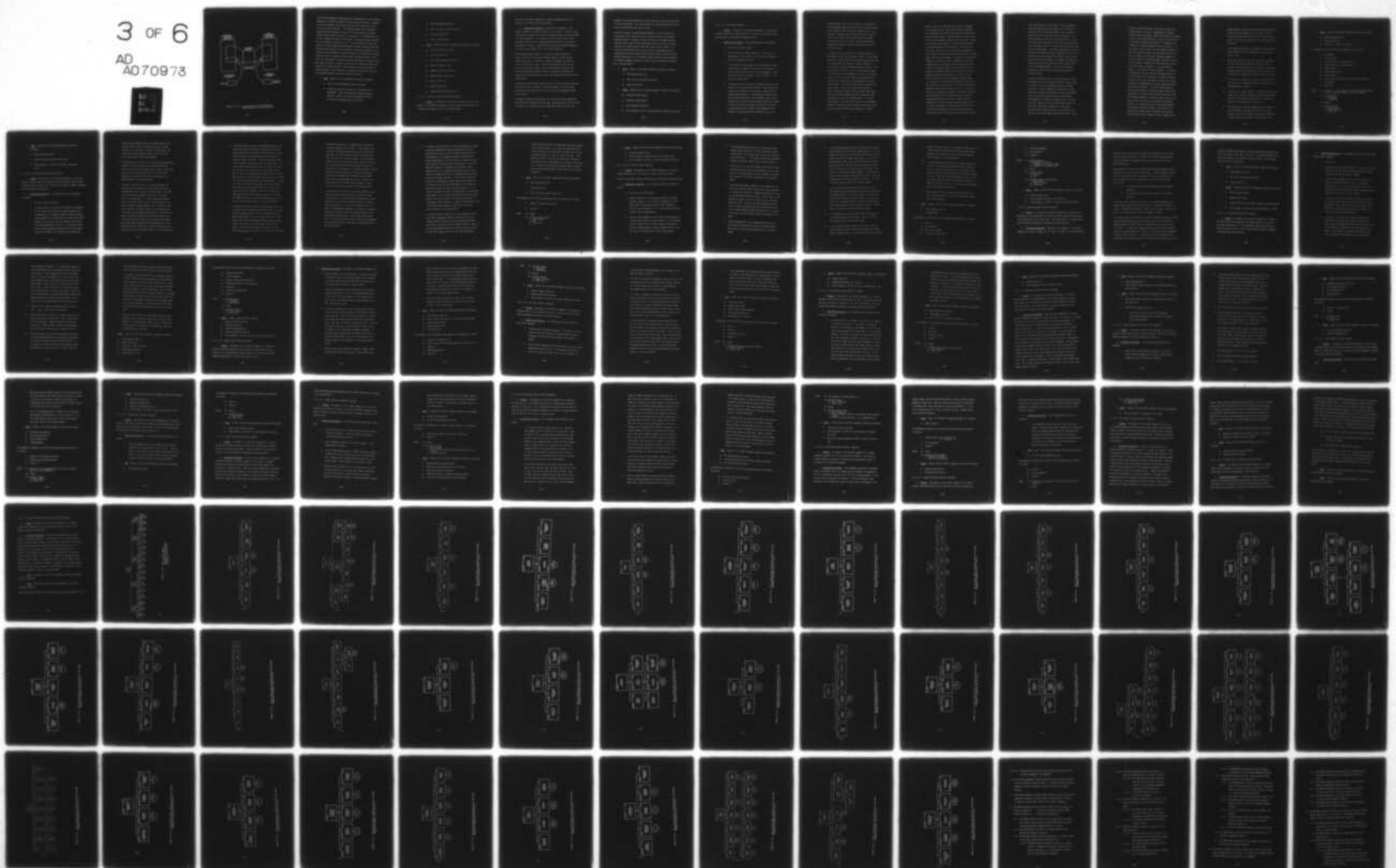
- Support Processing Component. This component performs such support functions as copying the data base from disk/core to tape, restoring the data base from tape to disk/core, removing incomplete updates when a PE aborts, and inspecting the data base for error conditions.

e. Inter-Complex Support. The DB Subsystem is designed to operate, to as great an extent as possible, within both the online Operational Complex (OPCX) and the offline Support Complex (SPCX). Figure 2.2.1-10 illustrates the transportability of a DB Subsystem module from one complex to the other.

AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955

UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

3 OF 6
AD
A070973



1.0 2.8 2.5 3.2 2.2 3.6 4.0 2.0 1.1 1.8 1.25 1.4 1.6

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

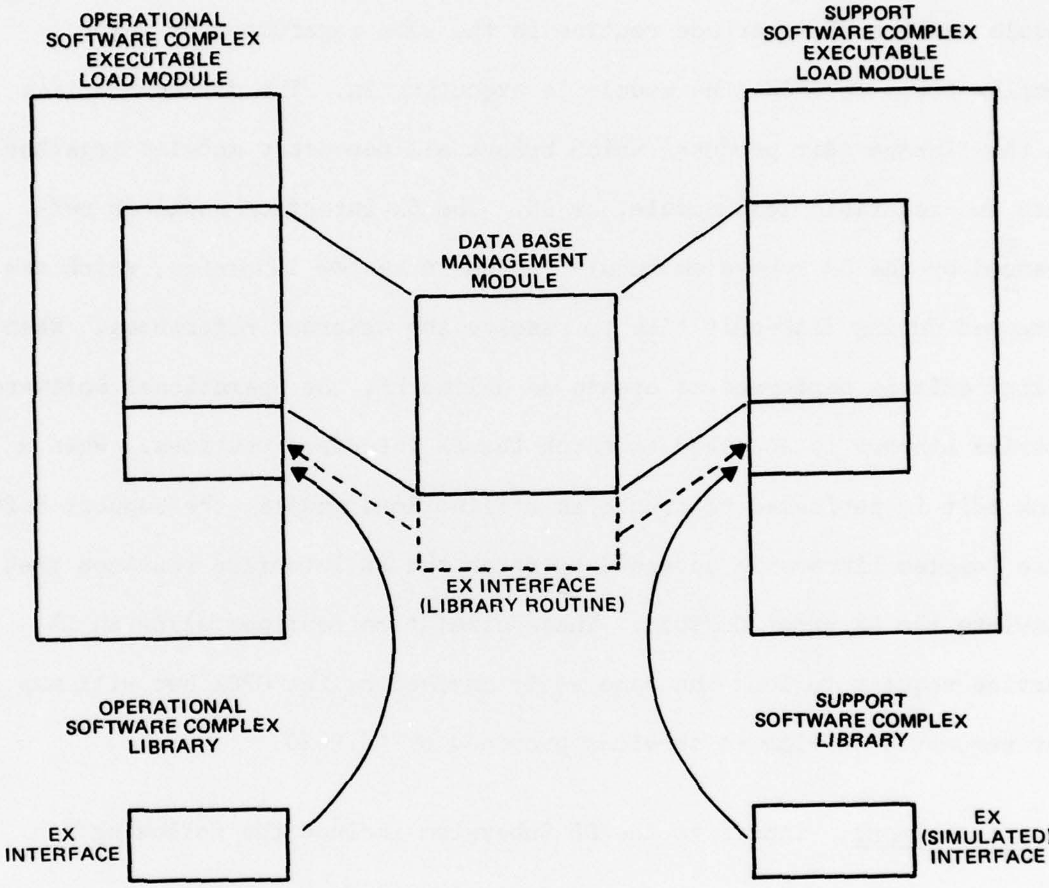


Figure 2.2.1-10. Transportability of a DB Subsystem Module Between the OPCX and the SPCX

The Data Base Management Subsystem module communicates with the Executive Subsystem (or OS/9020) through EX interface library routines. Examples of such EX interface routines are data base read, write, lock, clear lock, and PE abort requests. The interface between the DB Subsystem module and the EX interface routine is the same regardless of which complex (OPCX or SPCX) the module is executing in. The difference lies in the linkage edit process, which brings all necessary modules together into an executable load module, or PE. The EX interface routines referenced by the DB Subsystem module reside in system libraries, which are accessed during link-edit time to resolve the external references. When a link edit is performed to create an online PE, the Operational Software Complex Library is accessed to fetch the EX interface routines. When a link edit is performed to create an offline load module, the Support Software Complex Library is accessed to fetch the EX interface routines that simulate the EX under OS/9020. These simulation routines allow an EX service request to look the same as if invoked by the OPCX but will map the requested service to services provided by OS/9020.

f. Input. Inputs to the DB Subsystem include the following:

- The OAG and Non-Scheduled Flight Record Sets
- Index sets required to support the data base access mechanisms. These sets include the Flight Accession Table Set (FATS), the Flight Index Table Set (FITS), and the Arrival/Departure Table Set (ADTS)

- The Table Mapping Table Set
- Disk- and core-resident table sets
- Data Base Backup Set
- After Look recordings

g. Output. Outputs from the DB Subsystem include the following:

- Data base backup tape sets
- After Look tape sets
- Data base performance statistics
- Requested flight records
- Requested table data entries
- Updated flight record sets
- Updated index sets
- Updated table sets
- Updated Table Mapping Table Set

2.2.1.3.1 Retrieval Processing (DB-RETV) Component

a. Purpose. The purpose of the DB-RETV Component of the DB Subsystem is to provide the mechanism for accessing the data base and for returning all requested data to the calling module.

Note that the DB-RETV Component is a logical component and does not represent an invocable data base primitive.

b. Functional Description. There are six components in the DB-RETV Component, as shown in Figure 2.2.1-11 (page 1 of figure).Flight information is acquired for the calling module by record (via the DB-GETR or DB-GETE Components). The logical keys used in the access of this information can be arrival terminal, departure terminal, both arrival and departure terminal, a specific ACID, an airline operator designator, or relative record number within the flight record set.

When accessing flight records via a specific airline operator or aircraft identifier, the Flight Accession Table Set (FATS) and the Flight Index Table Set (FITS) are scanned to determine the location of the desired flight record. When accessing flight records via an arrival or departure terminal or an arrival/departure terminal pair, the Arrival/Departure Table Set (ADTS) is accessed to determine the location of the flight record set data block, which contains the desired flight record.

The relative record number, supplied either in the input calling sequence or from the FITS or ADT entry, is used to compute the relative block number within the flight record set and the record displacement within the data block.

Performance statistics pertaining to all data base retrieval operations are collected and logged on the ALQ tape. Any inconsistency in the format or content of the retrieved data is immediately reported via an error

message to the System Monitoring Position (SMP) and an error code returned to the calling module. The System Monitor can then take corrective action, such as reloading the data base from tape.

During the course of input message processing, some OPCX modules must have access to data base table sets to properly execute their message-handling function. Examples of such sets would be the Normal Capacity Table Set (NCTS) which contains information on the capacities of pacing airports; the Airline Operator Table Set (AOTS), which contains a list of valid air carrier/air taxi codes and their corresponding internal representation; and the Normal General Aviation Table Set (NGATS), which contains information on the estimated volume of general aviation flights. The DB-TABT Component performs the table retrieval function as requested by the calling module.

c. Input. Inputs to the DB-RETV Component include the following:

- Table Mapping Table Set
- Disk- and core-resident table sets
- Flight record sets

d. Output. Outputs from the DB-RETV Component include the following:

- Retrieved flight records
- Retrieved table entries
- Data retrieval statistics
- Error message to SMP if a data base error condition was found

2.2.1.3.1.1 Get Record (DB-GETR)

a. Purpose. The purpose of the DB-GETR Component is to retrieve a flight record from a flight record set. Note that the DB-GETR Component represents an invocable data base primitive.

b. Functional Description. The following steps are performed:

1. Log time of the GETR request

2. Validity-check the calling sequence. If an error is encountered, set the error code in the calling sequence, and send a message to the SMP.

3. If the relative record number is in the calling sequence, compute the block number and record position within the block (based on values found in the Table Mapping Table). Bypass the index scan process. Save the calling sequence record in DB work area

4. If flight record set access is by terminal (this is determined by the calling sequence content: arrival/departure terminal pair specified, or the arrival terminal with no flight number specified, or the departure terminal with no flight number specified), initiate DBGIND to get arrival terminal record from the Arrival/Departure Table Set (ADTS). If only the departure terminal is specified in the GETR calling sequence, set the GIND calling sequence arrival terminal to the provided arrival terminal code or the

lowest terminal code -1 if not provided. Set the GIND calling sequence departure terminal to the provided departure terminal code or the lowest terminal code -1 if not provided. The value of the data-item found in the ADT entry is the relative block number within the flight record set that is to be read. Save the GETR calling sequence and the ADTS record and positional pointers in the DB work area

5. If flight record set access is by flight number (this is determined by the calling sequence content: flight number specified or flight number with arrival terminal specified, or flight number with departure terminal specified , or flight number and both terminals),initiate DB-GIND. GIND will initiate DB-TABT to get the Flight Accession Table Set (FATS) record corresponding to the input OAG operator code or non-OAG call sign. Next it will scan the FATS record for the range entry that encompasses the input flight number. The result of the scan is a pointer that can be used as a key to the Flight Index Table Set. Next it initiates DB-TABT to get the Flight Index Table Set (FITS) record containing the first flight leg entry corresponding to the input flight number. If no such entry exists, a "no entry found" return code is given by DB-TABT. In this case, the "no entry found" return code will be set in the GETR calling sequence, a statistic

record is sent to the ALQ tape, and control is returned to the calling module. When a FITS record is returned, it is scanned for the first flight leg entry that matches the input flight number. When the entry is found, a check of the calling sequence is made to see if arrival or departure terminal was specified. If specified, the terminal ID is matched against the FITS entry. Note that one of the primary functions of the DB interface component is to filter flight records based on any given selection criteria input from the calling module. In order for the interface component to perform the filter operation, it must have a flight record to work with. When accessing the flight record set by flight number, the DB-GETR primitive can perform a preliminary filter function based on arrival or departure terminal without having to read a block of flight records, scan for the requested flight record, or transfer it to the interface component's work area as the FITS entry for every flight leg contains the arrival/departure terminal pair for that leg. If, in this case, an arrival or departure terminal is specified in the GETR calling sequence, and the terminal does not match the corresponding terminal in the FITS flight leg entry, continue the scan for the next flight leg entry (if any) for this flight number. The block header for the FITS record being scanned has an entry giving the number of flight leg entries within the block and an entry pointing

to an overflow block if one exists. If, in scanning for a matching flight leg entry, the end of FITS record is detected, check the overflow pointer. If an overflow block exists, it is read and the scan continues. If no matching flight leg entry is found after the scan, set a "no entry found" return code in the GETR calling sequence, send a statistic record to the ALQ tape, and return control to the calling module. If a flight leg entry is found that satisfies the selection criteria available at this level of the process, use the flight record pointer found in that entry (relative record number) to compute block number and record position within the block (based on values found in the Table Mapping Table). Save the GETR calling sequence, the FITS record, and positional pointers in the DB work area.

6. If flight record set access is by airline operator (this is determined by calling sequence content: airline operator alone, or airline operator with arrival terminal specified, or airline operator with departure terminal specified), initiate DB-GIND which will initiate DB-TABT to get the Flight Accession Table Set (FATS) record corresponding to the input OAG operator code or non-OAG call sign. Next, set the FATS record index to point to the first flight number range entry in the record. From this entry, obtain the pointer value that is to be used as a key to the Flight Index Table Set (FITS). Next,

GIND will initiate DB-TABT to get the FITS record containing flight leg entries corresponding to the input airline operator and range. When a FITS record is returned, a check of the calling sequence is made to see if arrival or departure terminal was specified. If specified, the terminal ID is matched against the first FITS entry. If the terminal does not match the corresponding terminal in the FITS flight leg entry, the next flight leg entry (if any) is obtained and checked for matching terminal ID. If, in scanning for a matching flight leg entry, the end of FITS record is detected, the overflow pointer in the FITS block header is checked. If an overflow block exists, it is read and the scan continues. If no match is found after the scan, then the FITS pointer from the next FATS range entry is obtained, DB-TABT is initiated to get the FITS record containing the next set of flight leg entries corresponding to the input airline operator, and the scan continues. If the last FATS range entry has been processed and no matching flight leg entry has been found, a "no entry found" return code is set in the GETR calling sequence, a statistic record is sent to the ALQ tape set, and control is returned to the calling module. If a flight leg entry is found that satisfies the selection criteria available at this level of the process, the flight record set pointer found in that entry (relative record number) is used to compute block

number (based on values found in the Table Mapping Table). The GETR calling sequence, the FATS record, the FITS record, and positional pointers in the DB work area are saved. The "Blocks Processed" bit map is initialized to all zeros.

7. If a flight record block is different from the requested block in memory, and it contains updates, write that block to disk.

8. Verify that the block just read is the block requested and that the format of the block header is valid. If an error condition exists, set error code in the calling sequence, send an error message to SMP, log a statistic record on the ALQ Tape Set and return error code to user.

9. If access is by terminal ID, the first flight record in the block for the requested terminal ID is moved to the calling module's work area.

10. If access is by flight number or by relative flight record number, then, using the computed record position within the block, the requested flight record is accessed and moved to the calling module's work area.

11. Set return code in calling sequence, send a statistic to the ALQ tape set, and return control to the calling module. Note that the completion statistic contains the location within the data base of the data base read.

c. Input. Inputs to the DB-GETR Component include the following:

- Table Mapping Table Set
- Flight record sets
- Index sets - FATS, FITS, and ADTS

The parameters in the DB-GETR calling sequence are as follows:

- Aircarrier[1]
- Flight number[1]
- Arrival Center or Pacing Airport[1]
- Departure Center or Pacing Airport[1]
- Record number[1]
- Flight Record Type[1,2]
- Access Method[3]
- Output Record Location (where record will be placed by DB-GETR)
- Status[3,4]

NOTES:

(1) Optional. If record number is given, flight record type must be given and parameters 1 through 4 ignored

(2) Possible values:
0 - Scheduled
1 - Nonscheduled

(3) Output

(4) Possible values:
0 - Record found
1 - Record not found
-# - Error code

d. Output. Outputs from the DB-GETR Component include the following:

- Retrieved flight record
- Record retrieval statistics on ALQ tape
- Error message to the SMP if any error condition was found

2.2.1.3.1.2 Get Next Record Component (DB-GETE)

a. Purpose. The purpose of the DB-GETE Component is to retrieve the next flight record that satisfies the identification criteria established by the last DB-GETR call. Note that the DB-GETE Component represents an invocable data base primitive.

b. Functional Description. The following steps are performed in DB-GETE:

1. Log time of the GETE request.
2. If flight record set access is by terminal (as determined by the content of the saved GETR calling sequence) and, if saved positional pointers indicate at least one flight record remains to be accessed from the flight record block currently in core, move the next available flight record to the calling module's work area. If the core resident block has been completely processed and the block header contains the address of an overflow block, compute the

relative record number of the first flight record in the overflow block and activate RFRB using relative record number as the access key. Transfer the RFRB return code to the GETE calling sequence, send a stat to ALQ tape, and return control to the calling module.

3. If flight record set access is by specific arrival/departure terminal pair and there are no more unprocessed flight records within the core-resident block or chain of overflow blocks, set the "EOF" return code in the GETE calling sequence, send a statistic record to ALQ tape set, and return control to the calling module.

4. If flight record set access is by arrival terminal and there are no more unprocessed flight records within the core-resident block or chain of overflow blocks, call GIND to set the departure terminal index equal to the next entry in the saved ADTS. If no more unprocessed entries exist in the ADTS record, set the "no entry found" return code in the GETE calling sequence, send a statistic record to the ALQ tape, and return control to the calling module. If at least one unprocessed entry exists in the saved ADTS record, convert the flight record set block number found in the next available ADTS entry to relative record number and activate RFRB using relative record number as the key. Transfer the RFRB return code to the GETE calling sequence, send a stat to ALQ, and return control to the calling module.

5. If flight record set access is by departure terminal and there are no more unprocessed flight records within the core-resident block or chain of overflow blocks, initiate GIND to get next arrival terminal record from Arrival/ Departure Table Set (ADTS). If no more unprocessed ADTS arrival terminal records exist, set the "EOF" return code in the GETE calling sequence, send a statistic record to the ALQ tape, and return control to the calling module. When at least one unprocessed ADTS arrival record is available, obtain the record via GIND, then set the record index to point to the matching departure terminal entry. Convert the flight record set block number found in the matching ADTS entry to relative record number and activate RFRB using relative record number as the key. Transfer the RFRB using relative record number as the key. Transfer the RFRB return code to the GETE calling sequence, send a stat to ALQ and return control to the calling module.

6. If flight record set access is by flight number,call GIND to scan the saved Flight Index Table Set (FITS) record for the next flight leg entry (if any) corresponding to the input flight number. If no such entry exists, set "EOF" return code in the GETE calling sequence, send a statistic record to ALQ tape, and return control to the calling module. When the entry is found, a check of the saved GETR calling sequence is made to see if arrival or departure

terminal was specified. If specified, the terminal ID is matched against the FITS entry. If the terminal does not match the corresponding terminal in the FITS flight leg entry, the scan continues for the next flight leg entry (if any) for this flight number. If FITS overflow blocks need to be read during the scan, they are read and the scan continues until either an entry is found or no more entries are available. If a flight leg entry is found that satisfies the selection criteria, then, using the relative record number from that entry, activate RFRB using relative record number as the key. Transfer the RFRB return code to the GETE calling sequence, send a stat to ALQ, and return control to the calling module.

7. If flight record set access is by airline operator and, if saved positional pointers indicate at least one flight record remains to be accessed from the flight record block currently in core, scan the flight records for a match on airline operator ID. If a match is found and there is a match on corresponding terminal ID (if arrival or departure terminal specified in saved GETR calling sequence), move the matching flight record to the calling module's work area, set record found return code in GETE calling sequence, send a statistic record to ALQ tape, and return control to the calling module.

8. If flight record set access is by airline operator and there are no more unprocessed flight records within the core-resident block, set a bit in the Blocks Processed bit map corresponding to block number processed, scan the saved Flight Index Table Set (FITS) record for a flight leg entry that matches on airline operator and possibly on arrival or departure terminal, if specified in saved GETR calling sequence record. If a matching flight leg entry is found, compute the flight record set block number from the pointer found in the flight leg entry. If the "Blocks Processed" bit-map bit corresponding to the computed block number is on, continue the scan of the FITS record for a flight leg entry that matches the input selection criteria. If the "Blocks Processed" bit-map bit corresponding to the computer block number is off, activate RFRB using the relative record number of the first record in the flight record block to be accessed as the access key. Transfer the RFRB return code to the GETE calling sequence, send a stat to ALQ, and return control to the calling module.

9. If flight record set access is by airline operator, there are no more unprocessed flight records within the core-resident block, and there are no more unprocessed flight leg entries within the saved FITS record, then, if there are no more unprocessed FATS entries for the airline operator, set the "EOF" return code in the GETE calling sequence,

send a statistic record to the ALQ tape, and return control to the calling module. If at least one unprocessed FATS entry is available, activate GIND to obtain from this entry the pointer value to be used as a key to the FITS. Next, initiate DB-TABT to get the next FITS record containing flight leg entries corresponding to the input airline operator. Process the FITS record, accessing flight record set blocks and returning matching flight records as previously described.

c. Input. Inputs to the DB-GETE Component include the following:

- Table Mapping Table Set
- Flight record sets
- The FATS, FITS, and ADTS index sets

The parameters in the DB-GETE calling sequence are described as follows:

- Address of data base work area
- Status[1, 2]

NOTES: (1) Output

(2) Possible return values:
0 - record found
4 - EOF
-# - Error code

d. Output. Outputs from the DB-GETE Component include the following:

- Retrieved flight record
- Record retrieval statistics written to the ALQ tape
- Error message to the SMP if an error condition was found

2.2.1.3.1.3 Get Table (DB-TABT) Component

a. Purpose. The purpose of the DB-TABT Component is to retrieve a table entry based on a key value or a relative position pointer.

Note that the DB-TABT Component represents an invocable data base primitive.

b. Functional Description. The following steps are performed in DB-TABT:

1. Log time of the TABT request.

2. Validity-check the calling sequence against the calling sequence template. If an error is encountered, set the error code in the calling sequence, send a message to the SMP, log the statistic on the ALQ tape set, and return control to the calling module.

3. Scan the Table Mapping Table Set (TMTS) pointer record for an entry which matches the input table ID. If no matching entry is found, set the "no table found" return code in the TABT calling sequence, send a statistic record to the ALQ tape, and return control to the calling module.

4. If a matching TMTS pointer entry is found, access the mapping record pointed to by the matching TMTS pointer record entry. If the target table is in core, access the first table block. If the target table is disk-resident, initiate a read SVC, which will get the table block.

5. If the table ID in the table header record does not match the table ID in the input calling sequence, set an error code in the calling sequence, send a message to the SMP, log a statistic on the ALQ, and return to the calling module.

6. If the calling sequence contains an entry number for the data record (Access Method Parameter in calling sequence = 1), compute the location of requested entry using a pointer in the Data Record Descriptor Group. If the table is disk resident, initiate a read SVC, which will get the table entry into core. Note that the data validation entries from the first table block are saved in the DB work area if the calling sequence indicates an update is to be done. If the table is core resident, access the requested table entry.

7. Move the table entry to the calling module's work area. Place size of returned table entry in calling sequence. Log a stat on the ALQ and return control to the calling module.

8. If the calling sequence contains an index into the pointer record (Table Set Access Mode in Mapping Record = 1), then, compute the location of the requested data record using the pointer found in the pointer table entry. If the table is disk resident, issue a read SVC which will get the table record into core. Note that the data validation entries from the first block are saved in the DB work area if an update is indicated by calling sequence. If the table is core resident, access the requested table entry. Move the table entry to the calling module's work area. Log a stat to ALQ and return control to the calling module.

9. If the calling sequence contains a key that is used to match against data records directly (Table Set Access Mode in Mapping Record = 0), search the table on a record-by-record basis looking for a match on key. If no match is found, set the "Table Entry Not Found" return code in the TABT calling sequence, send a statistic record to the ALQ tape, and return control to the calling module.

10. If a matching data record is found, move the table entry to the calling module's work area. Log a stat to ALQ and return control to the calling module.

11. If the calling sequence contains a key that is used to match against the Table Pointer Record (Table Set Access Mode in Mapping Record = 1), scan the entries in the pointer

record for a match on key. If no match is found, set the "Table Entry Not Found" return code in the TABT calling sequence, send a statistic record to the ALQ tape, and return control to the calling module.

12. If a matching pointer record entry is found, compute the location of the requested data record using the pointer found in the pointer table entry. If the table is disk resident, issue a read SVC which will get the table record into core. Note that the data validation entries from the first table block are saved in the DB work area prior to reading of the data record. If the table is core resident, access the requested table entry.

13. Move the table entry to the calling module's work area. Send a statistic record to the ALQ tape and return control to the calling module.

c. Input. Inputs to the DB-TABT Component include the following:

- Table Mapping Table Set
- Requested table

The parameters in the DB-TABT calling sequence are described as follows:

- Table ID
- Access Method[1]
- Key or entry number
- Location to receive entry

- Entry size expected
- Lock indicator[3]
- Status[2, 4]

NOTES: (1) Possible values:
0 = Parameter 3 is a key
1 = Parameter 3 is an entry number
2 = Parameter 3 is internal code

(2) Output

(3) Possible values:
0 = No lock
1 = Lock

(4) Possible status values returned:
0 - Record found
1 - Record not found
-# - Error code

d. Output. Outputs from the DB-TABT Component include the following:

- Retrieved table entry
- Table retrieval statistics on ALQ tape set
- Error message to the SMP if an error condition was found

2.2.1.3.2 Update Processing (DB-UPD) Component

a. Purpose. The purpose of the DB-UPD Component is to provide the mechanism for the modification of existing data base data and the insertion of new information into the data base. Note that DB-UPD is a logical component and does not represent an invocable data base primitive.

b. Functional Description. There are five components in the DB-UPD Component, as shown in Figure 2.2.1-11 (page 1 of figure). Flight information

may be modified (via the DB-CHGR Component), or added to the data base (via the DB-INST Component). A flight record may be logically removed from the data base by changing the status indicator to two (via the DB-CHGR Component).

When inserting a flight record via the DB-INST logical component, index pointers to this new record are established. All block headers accessed in the process of the update operation, and all data presented by the calling module as insert or change items, are checked for validity. This is done

- To ensure that the accessed data have not been damaged in any way
- To ensure that insert or change data do not damage the data base

Before any update operation takes place, the Data Base Management Subsystem ensures that the calling load module is authorized to request the update. An unauthorized attempt to update the data base is rejected and results in the generation of an error message to the SMP.

The Landing Capacities Set (CAPS) and General Aviation Estimates Set (GAES) input messages require an update operation against the User-Supplied Capacity Table Set (UCTS) and the User-Supplied General Aviation Table Set (UGATS). The Simulation Subsystem must perform an update operation against the Continue Table Set (CONTS). The SPCX must have the capability to modify various table sets. The logical data base component DB-SETT is called by the appropriate module to perform this update function. The

same data validation and security checking functions are performed here as in the update of the flight record sets. The DB-CTBE Component provides the capability for creating a new table entry or for deleting a table entry from the data base.

c. Input. Inputs to the DB-UPD Component include the following:

- Table Mapping Table Set
- Disk- and core-resident table sets
- Flight record sets

d. Output. Outputs from the DB-UPD Component include the following:

- Updated flight records
- Updated table entries
- Updated table sets with either inserted or deleted entries
- Error message to the SMP if an error condition is found

2.2.1.3.2.1 Insert Record (DB-INST) Component

a. Purpose. The purpose of the DB-INST Component is to insert a new flight record into either the OAG Flight Record Set or the Non-Scheduled Flight Record Set. The appropriate index set pointers are created at this time also. Note that the DB-INST Component represents an invocable data base primitive.

b. <u>Functional Description</u>. The following steps describe the steps in the DB-INST Component:

1. Start time.

2. Validity-check the calling sequence. If an error is encountered, set an error code in the calling sequence, send a message to the SMP, and return to caller.

3. Using the Table (Number) from the TMTS entry,check the update-authorization mask for a one. If the update-authorization bit is not on for the flight record set to be updated, set an error code in calling sequence, send a message to the SMP, and return to caller.

4. If the PE is authorized to insert a flight record, check the flight record block header to see if enough space is available to insert the new record in the current block. Note that the flight record block has been brought into core by a previous DB-GETR execution.

5. If there is not enough room to insert the new flight record into the current core-resident flight record block, read the housekeeping record for the flight record set and extract the pointer to the next available free block. If the number of remaining free blocks is less than the minimum as established by a System Parameter,

send a message to the SMP. If no free blocks remain for this flight record set, send a message to SMP, and return to the caller. Write a Before Look of the block header in core. Add the block number of the free block to the block header in core. Write an After Look of the updated block header. Issue a write SVC to write the core block back to disk. Issue a read SVC for the disk free block. Write a Before Look of the free block header. Update the free block header with the arrival/departure terminals and backward pointer to the block that overflowed. Write an After Look of the updated header of the overflow block.

6. Locate the first unused or deleted record in the core block. Write a Before Look of this record.

7. Move the new flight record to the flight record block, and write an After Look of the record. Write a Before Look of the block header. Update the number of filled and active slots. Write an After Look of the block header. Write the current block back to the flight record set.

8. Initiate DB-TABT to get the arrival terminal record from the Arrival/Departure Table Set (ADTS) that corresponds to the arrival terminal of the new flight record. If the departure terminal entry within the arrival terminal record does not have a pointer to the flight record set,

put the address of the newly acquired flight record block into the entry and initiate a write SVC for the ADT set.

9. Call PTRM to initiate DB-TABT to get the Flight Accession Table Set (FATS) record corresponding to the OAG operator code or Non-OAG call sign for the inserted record. Next, scan the FATS record for the range entry which encompasses the flight number of the inserted flight record. If the matching FATS entry does not contain a pointer to the Flight Index Table Set (FITS), obtain a block from the FITS free chain in the same manner as obtained for the flight record set and put the address of the newly acquired FITS block into the FATS entry by initiating a write SVC for the FATS table.

10. Initiate DB-TABT to get the Flight Index Table Set (FITS) block that is to contain the new flight leg entry corresponding to the just-inserted flight record. The new FITS flight leg entry is formatted and inserted and a write SVC is initiated for the FITS set.

c. Input. Inputs to the DB-INST Component include the following:

- Table Mapping Table Set
- Flight record sets
- Index Sets - FATS, FITS, ADTS
- Housekeeping Record Set
- Before/After-Look Set

The parameters in the DB-INST calling sequence are described as follows:

- Flight Record Type[1]
- Airline Operator
- Arrival Center or Pacing Airport
- Destination Center or Pacing Airport
- Aircraft ID
- Location of Flight Record
- Status[2, 3]

NOTES: (1) Possible values:
0 - Scheduled
1 - Nonscheduled

(2) Output

(3) Possible returns:
0 = Update complete
-# = Error code

d. Output. Outputs from the DB-INST Component

- Updated flight record sets
- Updated index sets
- Updated Housekeeping Set
- Update Before/After-Look Set
- Error message to the SMP if an error condition was found

2.2.1.3.2.2 Change Record (DB-CHGR) Component

a. Purpose. The purpose of the DB-CHGR Component is to update or delete an existing flight record that resides in either the OAG Flight Record Set or the Nonscheduled Flight Record Set. Note that the DB-CHGR Component represents an invocable data base primitive.

b. Functional Description. The steps in the DB-CHGR Component are:

1. Validity-check the calling sequence. If error encountered, set an error code in the calling sequence, send a message to the SMP, release any data base locks initiated by this PE, log the error statistic, and abort the PE.

2. Using the Table Number from the TMTS entry, check update-authorization mask for a one. If update-authorization bit is not on for the flight record set to be updated, set an error code in the calling sequence, send a message to the SMP, and return to caller.

3. If the PE is authorized to update a flight record, access the item in the calling sequence that contains the relative record position. Using this position, compute the relative block number and record position within the flight record block. Note that the flight record block that is to contain the updated record must be core resident at this time as the result of a just-completed execution of DB-GETR. If the block number computed from the updated record does not match the core-resident block, set an error code in the calling sequence, send a message to the SMP, and return to caller.

4. Given that the PE is authorized to update a flight record and the proper flight record block is core resident, then

write a Before Look of the record to be updated. Move the updated flight record from the calling module's work area to its proper position within the flight record block. Write an After Look. If the update was a delete, write a Before Look of the header, update the header and write an After Look. Activate GIND to locate the FIS entry for the deleted record and then activate UFIS to remove the pointer to the deleted flight record.

5. Write the flight record block to disk, set the "Record Changed" return code in the calling sequence, and returr control to the calling module.

c. Input. Inputs to the DB-CHGR Component include the following:

- Table Mapping Table Set
- Flight record block which is to contain updated record
- Updated flight record
- Before/After-Look Set
- DB Work Area Set

The parameters in the DB-CHGR calling sequence are described as follows:

- Location of updated record
- Record location within calling module's work area to be updated
- Flight Record Type[1]
- Status[2, 3]

NOTES: (1) Possible values:
0 - Scheduled
1 - Nonschedules

(2) Output

(3) Possible Returns:
0 = Update successful
-# = Error code

d. Output. Outputs from the DB-CHGR Component include the following:

- Updated flight record sets
- Updated Before/After-Look Set
- Error message to the SMP if an error condition was found

2.2.1.3.2.3 Set Table (DB-SETT) Component

a. Purpose. The purpose of the DB-SETT Component is to update an existing disk- or core-resident table entry. Note that the DB-SETT Component represents an invocable data base primitive.

b. Functional Description. The following steps are performed in the DB-SETT Component:

1. Validity-check the calling sequence. If an error is encountered, set an error code in the calling sequence, send a message to the SMP, and return control to the calling module.

2. Using the Table (Number) from the TMTS entry, check the update-authorization mask for a one. If the update-authorization bit is not on for the flight record set to be updated, set

an error code in calling sequence, send a message to the SMP, and return to caller.

3. If the PE is authorized to update the table set, and if the key items as given in the saved DB-TABT calling sequence, set an error return code in DB-SETT calling sequence, and return control to the calling module.

4. If the PE is authorized to update the table set, use the data-item validation entries from the table's Data Record Validation Record, to validate data-items in the updated table entry as to format and minimum and maximum values. If any data-item does not pass the validity check, set an error code in the calling sequence, send a message to the SMP, and return control to the calling module.

5. If all data-items in the updated table entry are valid and the table is core resident, write a Before Look of the old record on the Before/After Look Tape, and move the entry to its proper location in the table. Write an After Look on the Before/After Look Tape. Then, using disk location information in the Table Mapping Table, read the block of the disk-resident version of the core table and write the updated table block to disk. Set the "Table Entry Replaced" return code in the calling sequence, and return control to the calling module.

6. If all data-items in the updated table entry are valid and the table is disk resident, write a Before Look of the old record, move updated entry to saved core-resident table block, and write an After Look. Next, write the table block to disk, set the "Table Entry Replaced" return code in the calling sequence, and return control to the calling module.

c. Input. Inputs to the DB-SETT Component include the following:

- Saved table block
- Updated table entry
- Saved DB-TABT calling sequence
- Table Mapping Table Set
- Before/After Look Tape Set

The parameters in the DB-SETT calling sequence are described as follows:

- Table ID
- Key Value
- Location of new entry
- Status[1, 2]

NOTES: (1) Output

(2) Possible values for the status variable:
0 = Update complete
-# = Error code

d. <u>Output</u>. Outputs from the DB-SETT Component include the following:

- Updated table set
- Updated Before/After Look Tape Set
- Error message to the SMP if an error condition was found

2.2.1.3.2.4 Create Table Entry (DB-CTBE) Component

a. <u>Purpose</u>. The purpose of the DB-CTBE Component is to insert a new entry into a core- or disk-resident table set. The appropriate table pointers are created or adjusted at this time if necessary. Note that the DB-CTBE Component represents an invocable data base primitive.

b. <u>Functional Description</u>. The following steps are performed in the DB-CTBE Component:

1. Initiate DB-TABT to determine if the entry to be inserted is already in the table. If an entry is returned by DB-TABT, set the "Duplicate Key Value" return code in the CTBE calling sequence, and return control to the calling module. If no entry is returned by TABT, search the table for first free space and insert the new entry. This may involve disk reads if primary table residence is on disk. Note that the TABT execution will have brought the first table block into core for any disk-resident table. Also note that most tables reside within one block. If the table contains more than two blocks, the table's pointer record is updated at this time by adding a pointer entry

containing the new key value and a pointer to the just-inserted table entry. If, at table-entry-insertion time, there is no room in an existing table block to insert the new entry, a free block is obtained (if available) for the insertion. If there is no room in the table to hold the new entry, set the "Entry Will Not Fit In Table Space" return code in the calling sequence, send a message to the SMP, and return control to the calling module. All updates are written to disk.

c. Input. Inputs to the DB-CTBE Component include the following:

- Table entry to be inserted
- Table Mapping Table Set
- Table which is to contain the new entry

The parameters in the DB-CTBE calling sequence are described as follows:

- Table ID
- Key value
- Location of entry
- Status[1, 2]

NOTES: (1) Output

(2) Possible values for the status variable
0 = Record inserted
-# = Error code

d. Output. Outputs from the DB-CTBE Component include the following:

- Updated table set
- Error message to SMP if an error is found

2.2.1.3.3 Initialize Data Base (DB-INIT) Component

a. Purpose. The purpose of the DB-INIT Component is to provide the mechanism for the initialization of the flight record sets, flight record index sets, and table sets and their logical addition or deletion from the data base. Note that DB-INIT is a logical component and does not represent an invocable data base primitive.

b. Functional Description. There are three components in the data base initialization component, as shown in Figure 2.2.1-11 (page 1 of figure). The flight record sets are initialized via the DB-CFRS Component. All record blocks within the flight record files are linked together as a chain of free blocks. A record in the Housekeeping Set (HKS) is created that points to the first block in the free chain. As data records are added to the flight record set,the Housekeeping Set Pointer Record is used to pull the next available block from the free chain. Flight Record and Table sets are initialized via the DB-CTAB Component. This component creates a Table Mapping Table Set entry for the new table and writes a header on each block in the table itself. Note that disk space for the flight record sets, flight record index sets, and table sets must have been allocated prior to execution of these initialization components. Table sets are logically removed from the data base via the DB-DTAB Component. This function consists of deleting the table's entry from the Table Mapping Table Set (TMTS).

c. Input. Inputs to the DB-INIT Component include the following:

- Table Mapping Table Set
- Noninitialized flight record set, flight record index sets, and table sets

d. Output. Outputs from the DB-INIT Component include the following:

- Initialized flight record sets, flight record index sets, and table sets
- Updated Table Mapping Table Set
  - New entries created by the DB-CTAB Component
  - Null entries (signifying table deletion) created by the DB-DTAB Component

2.2.1.3.3.1 Create Flight Record Set (DB-CFRS) Component

a. Purpose. The purpose of the DB-CFRS Component is to load the scheduled flight record set and create all indices. Note that the DB-CFRS Component represents an invocable data base primitive.

b. Functional Description. The following steps describe the component:

1. Buffer flight records sorted by aircraft ID until the airline operator changes or the count of flight records reaches fifty and the aircraft ID changes.

2. Store each flight record in the next available slot of the flight record block for this arrival departure pair. If no block exists for this arrival departure pair, one is created from the Flight Record Set Free Chain. The Housekeeping Set is then updated to point to the next free block. If the existing block for this arrival departure pair is full, an overflow block is created from the Flight Record Set Free Chain. The Housekeeping Set is then updated to point to the next free block.

3. The standard header in the flight record block is updated.

4. The arrival departure table is updated, if for this arrival departure pair there is no pointer to the flight record set. The Housekeeping Set is then updated to point to the next free block in the flight record set.

5. Each buffer of flight records updates a single FIS record. The Housekeeping Set is then updated to point to the next FIS record. When a FIS block gets full, the next one is read in and the Housekeeping Set is updated to point to the next free block. While creating the flight record set there will be no overflow in the FIS.

6. The standard header in the FIS block is updated.

7. The FAT is updated indexed by airline operator.

8. The standard header in the FAT block is updated.

c. Input. Inputs to the DB-CFRS Component include the following:

- Noninitialized flight record set
- Table Mapping Table Set
- System Parameter Set
- Housekeeping Set

The parameters contained in the DB-CFRS calling sequence are described as follows:

- Address of a flight record
- Status[1]

NOTES: (1) Possible values:
0 = Good
-# = Error code

d. Output. Outputs from the DB-CFRS Component include the following:

- Initialized flight record set
- Updated Housekeeping Set

2.2.1.3.3.2 Create Table (DB-CTAB) Component

a. Purpose. The purpose of the DB-CTAB Component is to initialize the table headers, the flight record headers, and to create a Table Mapping Table Set (TMTS) entry for each. Note that the DB-CTAB Component represents an invocable data base primitive.

b. Functional Description. The following steps describe the component:

1. Using the table set default values from the System Parameter Set, build and write the table header for each block to disk. If the table is to contain a chain of free blocks, the Housekeeping Set record is written at this time which points to the first block in the free chain.

2. Next, a Table Mapping Table Set (TMTS) entry is generated describing the characteristics of the new table set. Set the "Table Created" return code in the calling sequence, and return control to the calling module.

c. Input. Inputs to the DB-CTAB Component include the following:

- Noninitialized table set
- Table Mapping Table Set
- System Parameter Set
- Housekeeping Set

The parameters contained in the DB-CTAB calling sequence are described as follows:

- Address of TAB Description Buffer[1]
- Address Data Validation Buffer[1]
- Status[2, 3]

NOTES: (1) Optional: If not supplied, then the value is obtained from the System Parameter Set

(2) Output

(3) Possible returns:
0 = Table created
-# = Error code

d. Output. Outputs from the DB-CTAB Component include the following:

- Initialized table set
- Updated Housekeeping Set
- Updated Table Mapping Table Set
- Error message to the SMP if an error condition is found

2.2.1.3.3.3 Delete Table (DB-DTAB) Component

a. Purpose. The purpose of the DB-DTAB Component is to logically delete a table from the data base by setting the Table Mapping Table (TMT) entry for the deleted table to null values. Note that the DB-DTAB Component represents an invocable data base primitive.

b. Functional Description. The following steps describe the components:

1. The Table Mapping Table Set is accessed and the status indicator for the deleted table is set to two. Then TMT update is also written to the disk-resident version of the TMT. Set the "Table Deleted" return code in the calling sequence, and return control to the calling module.

c. Input. Input to the DB-DTAB Component includes the following:

- Table Mapping Table Set

The parameters contained in the DB-DTAB calling sequence are described as follows:

- Table ID
- Status[1, 2]

NOTES: (1) Output

(2) Possible returns:
0 = Table deleted
-# = Error code

d. Output. Outputs from the DB-DTAB Component include the following:

- Updated Table Mapping Table Set
- Error message to the SMP if an error condition is found

2.2.1.3.4 Support Processing (DB-SPRT) Component

a. Purpose. The purpose of the DB-SPRT Component is to provide the support mechanism required to perform data base startup, recovery, backup, and inspection operations. Note that DB-SPRT is a logical component and does not represent an invocable data base primitive.

b. Functional Description. There are four components in the Support Processing Component, as shown in Figure 2.2.1-11 (page 1 of figure). The data base backup process (performed by the DB-BKUP Component) is initiated by an operator keyin. While the backup operation is in progress, all data base updates are inhibited. All dynamic core-resident table and index sets are written to disk followed by a write of all disk-resident table sets, flight record index sets, and flight record sets to tape. The

data base startup process (performed by the DB-STUP Component) is initiated by the EX Subsystem.

2.2.1.3.4.1 Backup Data Base (DB-BKUP) Component

a. Purpose. The purpose of the DB-BKUP Component is to copy the data base from disk/core to tape. Note that the DB-BKUP Component represents an invocable data base primitive which is initially activated by an operator keyin.

b. Functional Description. The following steps describe the component:

1. Set the data base lock to inhibit data base updates during the backup operation. Write the backup tape numbers to the Recovery/Restore Set (RSET).

2. Send a broadcast message to the system terminals - "Data Base Backup in Progress".

3. Scan the Table Mapping Table Set (TMTS) looking for mapping records which point to core-resident table sets and index sets. Whenever such a mapping record is found, the core-resident table or index pointed to is written to disk.

4. Using the TMTS to supply the list of data base sets, write all sets from disk to the backup tape. When all data base sets have been written to tape, then send a broadcast message to the System Terminals of "Data Base Backup Complete".

Then release the data base lock, set the "Copy Complete" return code in the calling sequence, send the completion statistics record to the SAR tape, and return control to the calling module.

c. Input. Inputs to the DB-BKUP Component include the following:

- DB-BKUP calling sequence
- CFC disk-/core-resident data base

The parameters contained in the DB-BKUP calling sequence are described as follows:

- Logical device number for the tape output unit
- Status[1, 2]

NOTES: (1) Output

(2) Possible returns:
1 - Copy complete
2 - Error in calling sequence; code indicates data-item in error
3 - Error in data base

d. Output. Outputs from the DB-BKUP Component include the following:

- Updated DB-BKUP calling sequence
- Data base backup statistics on the SAR tape
- Tape copy of the CFC data base
- "Data Base Backup in Progress" broadcast message
- "Data Base Backup Complete" broadcast message

2.2.1.3.4.2 Data Base Startup (DB-STUP) Component

a. Purpose. The purpose of the DB-STUP Component is to bring the CFC data base up to full operational status by loading either from tape to disk/core or from disk to core, depending on the operator option selected. Note that the DB-STUP Component represents an invocable data base primitive which is initially activated by an operator keyin.

b. Functional Description. The following steps describe the component:

1. If the startup option specifies disk to core, load the Table Mapping Table Set (TMTS) into core from disk. Scan the TMTS to look for mapping records which point to table sets and index sets that are designated as core resident. Whenever such a mapping record is found, the disk-resident table or index pointed to is read into core. When all designated table sets and index sets are loaded to core, set the "Startup Complete" return code in the calling sequence, send the completion statistic record to the SAR tape, and return control to the calling module.

2. If the startup option specifies tape to disk/core, then read the data base sets from the backup tape and write them to disk. Next load the TMTS into core from disk. Scan the TMTS looking for mapping records which point to table sets and index sets that are designated as core resident. Whenever such a mapping record is found, the disk-resident

table or flight record pointed to is read into core. As each table set is loaded to core, it is validated as to format by checking each data-item against the specifications set forth in the Standard Table Validation Record for that table set. The disk-resident table sets are read into core, one block at a time, and are validated in the same manner as the core-resident table sets and index sets. If any data-items are found to be in error, then the "Data Base Error" flag is set and the remainder of the core-resident data is loaded without validity checking (if not already loaded), the "Error in Data Base Copy" return code is set in the calling sequence, an error message is sent to the SMP, an error statistic record is written to the SAR tape, and the process is halted. The operator at the SMP at this point has the option to restart the data base startup operation or to restart the data base startup operation with another data base version.

3. If all table sets and core-resident flight record index sets are valid, then validate all flight record set index pointers by looping through the Airline Operator Table (via TABT) and access flight records (via GETR/GETE) using airline operator as key.

4. When all flight index set pointers have been validated, access the Recovery/Restore Set (RSET) and scan for an entry which contains the same tape ID as the data base

backup tape used to load this version of the data base. If no matching entry is found or if the matching entry contains no SAR tape numbers, then set the "Startup Complete" return code in the calling sequence, write the completion statistic record to the SAR tape, and return control to the calling module.

5. If a matching entry is found which contains SAR tape IDs, then, for each SAR tape listed in the entry, request operator mount of the SAR tape, scan the tape for data base update statistic records and initiate all data base updates using data base primitives. When all updates from all SAR tapes have been processed, set the "Startup Complete" return code in the calling sequence, write the completion statistic record to the SAR tape, and return control to the calling module.

c. Input. Inputs to the DB-STUP Component include the following:

- CFC tape-resident data base
- SAR tapes associated with the tape-resident data base
- CFC disk-resident data base

The parameters contained in the DB-STUP calling sequence are described as follows:

- Data base copy identification[1]
- Startup option[2]
- Status[3, 4]

NOTES: (1) Not necessary if startup option = 1

(2) Possible values:
0 - Tape to disk/core
1 - Disk to core

(3) Output

(4) Possible return values:
1 - Startup complete
2 - Error in calling sequence; code indicates the data-item in error
3 - Error in the data base copy or the transaction file

d. Output. Outputs from the DB-STUP Component include the following:

- Disk/core resident data base
- Error message to the SMP if a data base error condition was found
- Data base startup completion statistic record on the SAR tape

2.2.1.3.4.3 Data Base Recovery (DB-RCOV) Component

a. Purpose. The purpose of the DB-RCOV Component is to remove incomplete updates from the data base when a PE aborts. Note that the DB-RCOV Component represents an invocable data base primitive which is activated by the Executive Subsystem.

b. Functional Description. This paragraph describes the component. Access the header record in the Before Look Set (BLOOKS) and compare the PE ID in the header record with the PE ID in the calling sequence. If the IDs do not match, then halt the process. If the IDs do match, access each block in the BLOOKS (the number of active locks is indicated in the

header record), and write these blocks back to the set ID that has been appended to each block. When all blocks have been written, then clear the BLOOKS header record and write it back to the BLOOKS Set. If the list of change blocks is in core, then clear the list. Return control to the Executive Subsystem.

c. <u>Input</u>. Input to the DB-RCOV Component includes the following:

- Before Look Set

The parameters contained in the DB-RCOV calling sequence are described as follows:

- Abort indicator $\begin{bmatrix} 0 = \text{System abort} \\ 1 = \text{PE abort} \end{bmatrix}$
- ID of aborted PE
- Status[1, 2]

NOTES: (1) Output

(2) Possible return values:
0 - Recovery is complete
1 - Error in the data base

d. <u>Output</u>. Outputs from the DB-RCOV Component include the following:

- Restored before looks
- Cleared Before Look Set

2.2.1.3.4.4 Inspect Data Base (DB-INSP) Component

a. <u>Purpose</u>. The purpose of the DB-INSP Component is to inspect either a given data base set or all data base sets and to indicate the

first error found with a message indicating the type of error found and the location of the data-item in error. Note that the DB-INSP Component represents an invocable data base primitive which is activated during startup or shutdown.

b. Functional Description. The following steps describe the component:

1. If an inspection of the Data Base is specified by the operator, the Data Base Inspect routine will systematically verify all linkages between associated Data Base files. The system operator may also specify a percentage of inspection to be performed, 100% being the default value. All data and file linkages will be verified for the specified portion of the data base.

c. Input. Input to the DB-INSP Component includes the following:

- CFC disk-/core-resident data base

The parameters contained in the DB-INSP calling sequence are described as follows:

- Set ID[1]
- % to be inspected
- Status[2, 3]

NOTES: (1) Optional--if not present, the entire data base will be inspected

(2) Output

(3) Possible return values:
0 = Inspection complete
-# = Error found

d. Output. Outputs from the DB-INSP Component include the following:

- DB error messages if discrepancies are found

2.2.1.3.5 Data Base Interface (DB-APIN) Component

a. Purpose. The purpose of the DB-APIN Component is to act as an interface between the OPCX message processor modules and the DB-GETR, DB-GETE, DB-INST, and DB-CHGR primitives by performing record filtering, data extraction, and array-processing operations. Note that the DB-APIN Component is a logical component and does not represent an invocable data base primitive.

b. Functional Description. There are two components in the DB-APIN Component, as shown in Figure 2.2.1-11(page 1 of figure). Flight information is acquired for the calling message processor module via the DB-GTFR Component. Using the input key, qualifier, and information parameters, retrieve flight records by initiating DB-GETR/DB-GETE. Filter the returned records based on qualifier information, extract data from the selected flight records based on the information parameters, and place it in array for output to the calling module. Flight record update or insertion is performed by the DB-UPFR Component. For flight record insertion requests, the DB-UPFR first ensures that a duplicate record already exists in the data base by activating DB-GETR. When it is determined that the flight record does not already exist in the data base,

the new flight record is formatted using information input by the calling module and DB-INST is activated to perform the actual record insertion. For flight record modification, DB-GETR is activated to get the record to be modified. When the proper record is obtained, the information passed by the calling message processor module is moved to the record, which is then written back to the Flight Record Set via the DB-CHGR primitive.

c. Input. Inputs to the DB-APIN Component include the following:

- Application Flight Record Retrieval Control Set (APFRRS)
- Application Update Control Set (APUCS)

d. Output. Outputs from the DB-APIN Component include the following:

- Filtered Flight Record Set (APFRDS)
- Status-indicator data-item

2.2.1.3.5.1 Get Flight Record Interface (DB-GTFR) Component

a. Purpose. The purpose of the DB-GTFR Component is to retrieve flight record information (as opposed to flight records) from the OAG Flight Record Set and the Nonscheduled Flight Record Set.

b. Functional Description. The DB-GTFR Component receives key, qualifier, and requested information parameters via calling sequence. If continuation is not indicated, validate the key parameters to ensure that they form a consistent set, and build a flight information extraction

table from the requested information parameters and a physical description of the logical flight record. Set up the parameter sequence for a get-record request setup, and invoke DB-GETR to obtain the first logical flight record satisfying the key parameters. If successful, filter this record based on the qualifier information provided in the calling sequence. If the record survives the filtering process, extract the requested information. Obtain the next record by invoking DB-GETE and perform the same process of filtering and extraction. Terminate this process (with appropriate status conditions) if any one of the following conditions occurs:

- An error status is returned from the data base primitives.
- The data base primitives indicate that there are no more records in the chain.
- The maximum output array space is reached.

If continuation is indicated and allowed (i.e., there has been a previous initial call), skip the setup and initial call to DB-GETR and proceed in the DB-GETE-filter-extract loop. If continuation is indicated but not allowed, indicate an error condition and return control to the calling module.

c. Input. The input to the DB-GTFR Component is Application Flight Record Retrieval Control Set (APFRRS).

d. Output. The output from the DB-GTFR Component is Filtered Flight Record Set (APFRDS).

2.2.1.3.5.2 Update Flight Record Interface (DB-UPFR) Component

a. Purpose. The purpose of the DB-UPFR Component is to change flight record information in an existing flight record or add or delete a flight record to the data base.

b. Functional Description. The DB-UPFR Component receives information via a calling sequence. If the request is for a record addition, set up a DB-GETR calling sequence, invoke DB-GETR, and filter any returned records to ensure that the record does not exist on the data base. If the request is for a change or delete, the key information must be a relative record number and set designator. Invoke DB-GETR to obtain the correct relative record. In either case, validate the information being placed in the data base. For an addition, build a logical flight record, ensuring that all necessary information is provided. For a change request, update the logical record retrieved. Invoke DB-INST to insert the record and invoke DB-CHGR to perform the change request.

c. Input. The input to the DB-UPFR Component is Application Update Control Set (UPUCS).

d. Output. The output from the DB-UPFR Component is the status-indicator data-item.

The Visual Table of Contents for the DB Subsystem is shown in Figure 2.2.1-11.

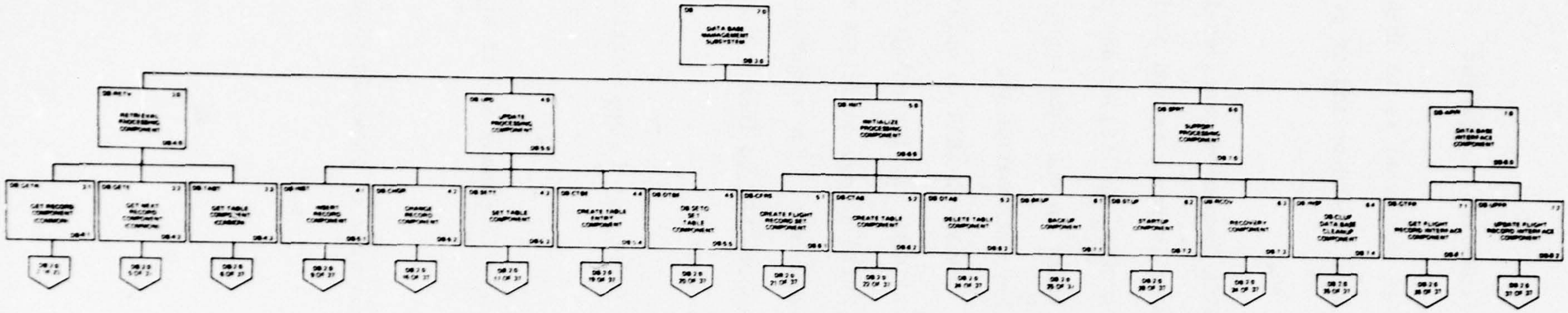


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (1 of 37)

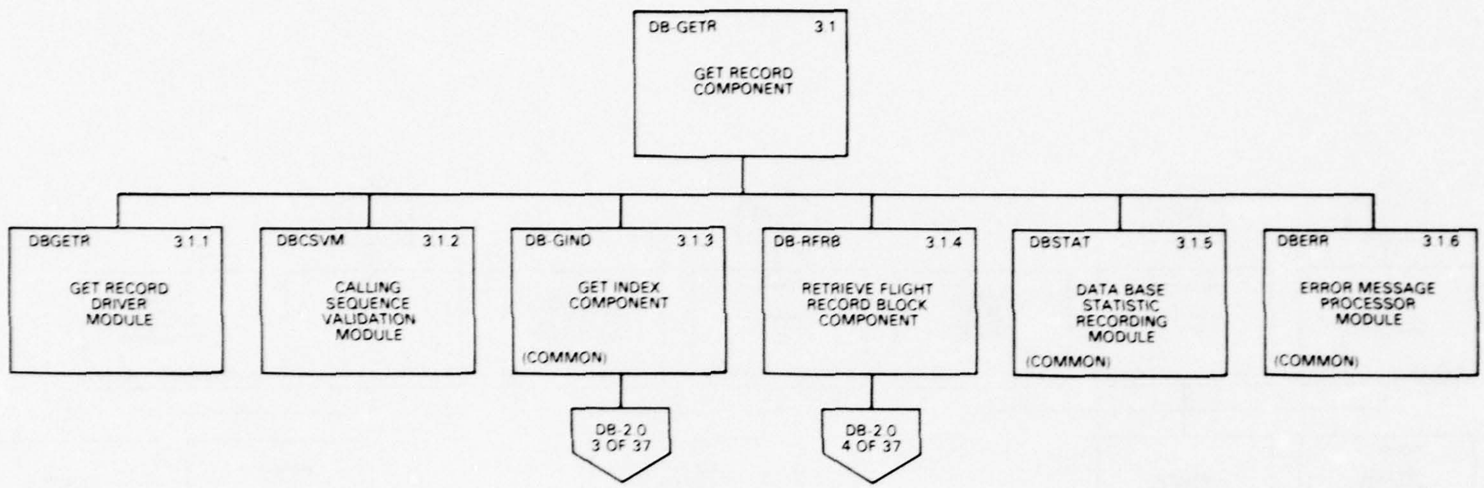


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (2 of 37)

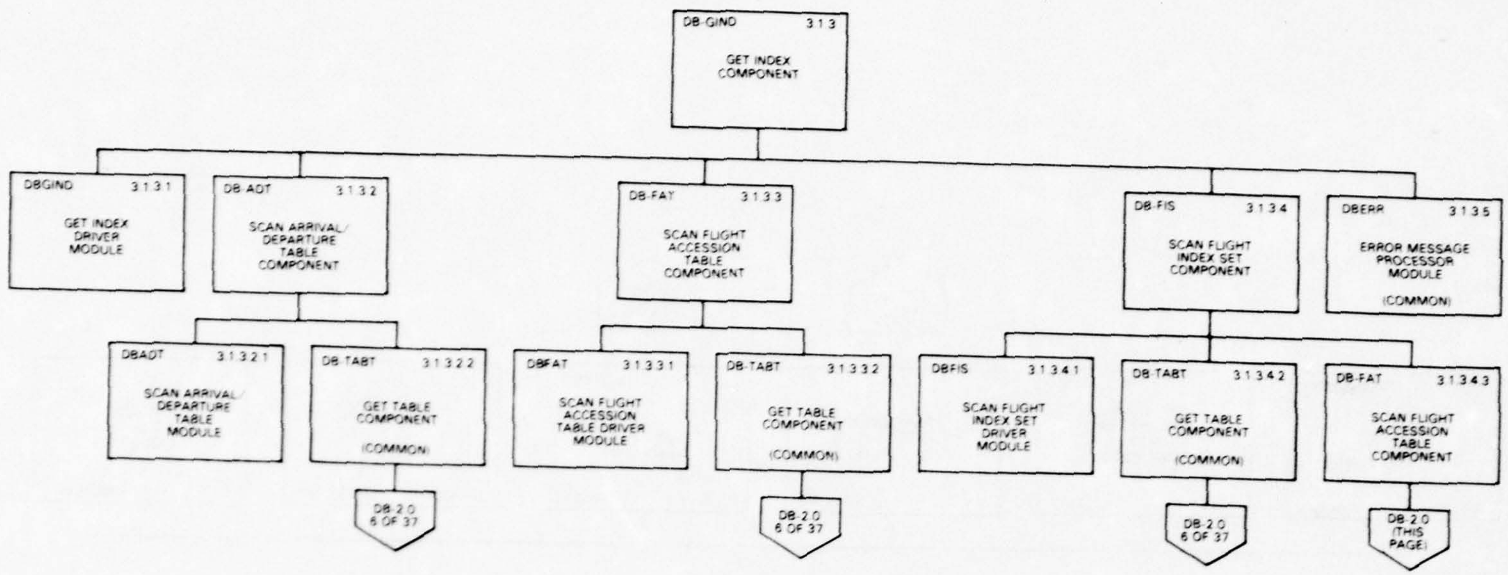


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (3 of 37)

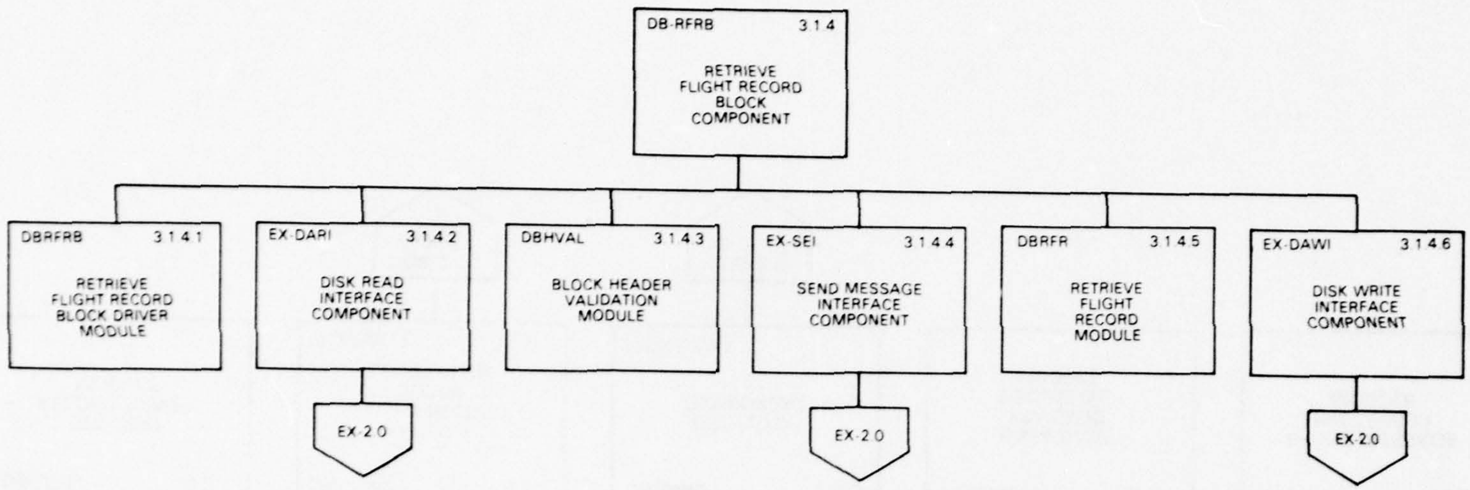


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (4 of 37)

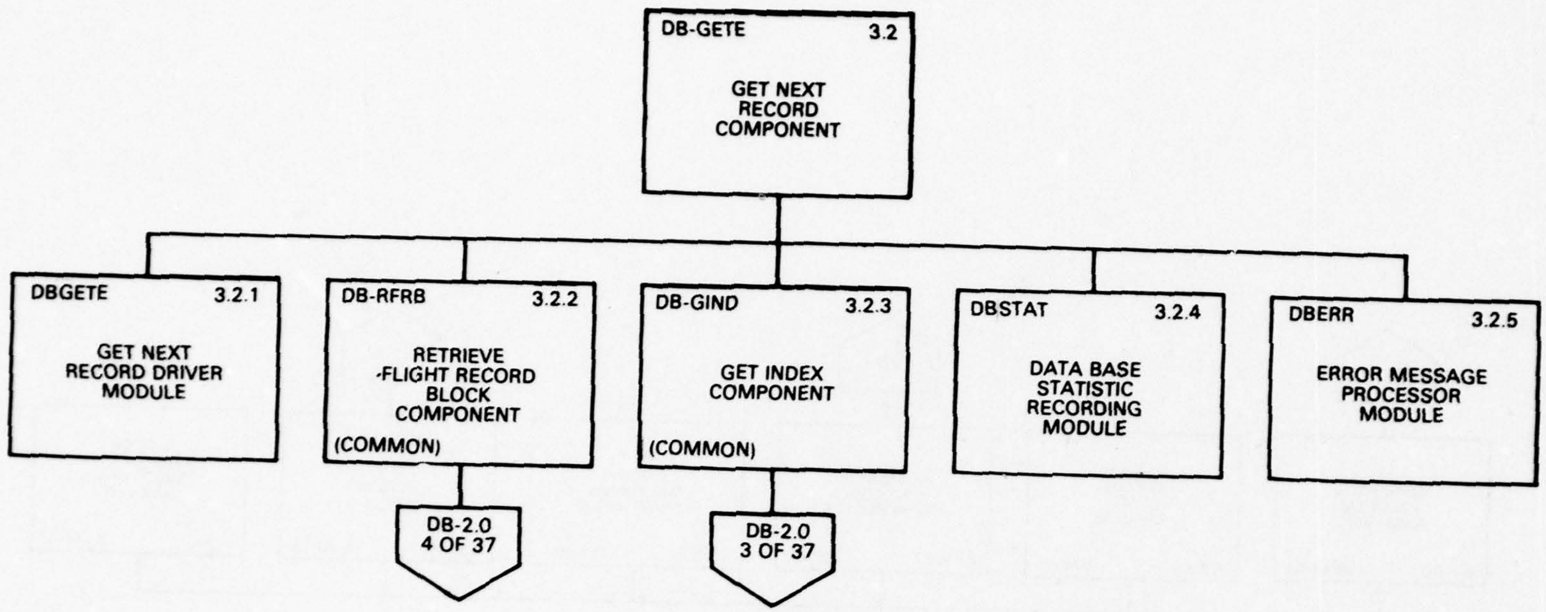


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (5 of 37)

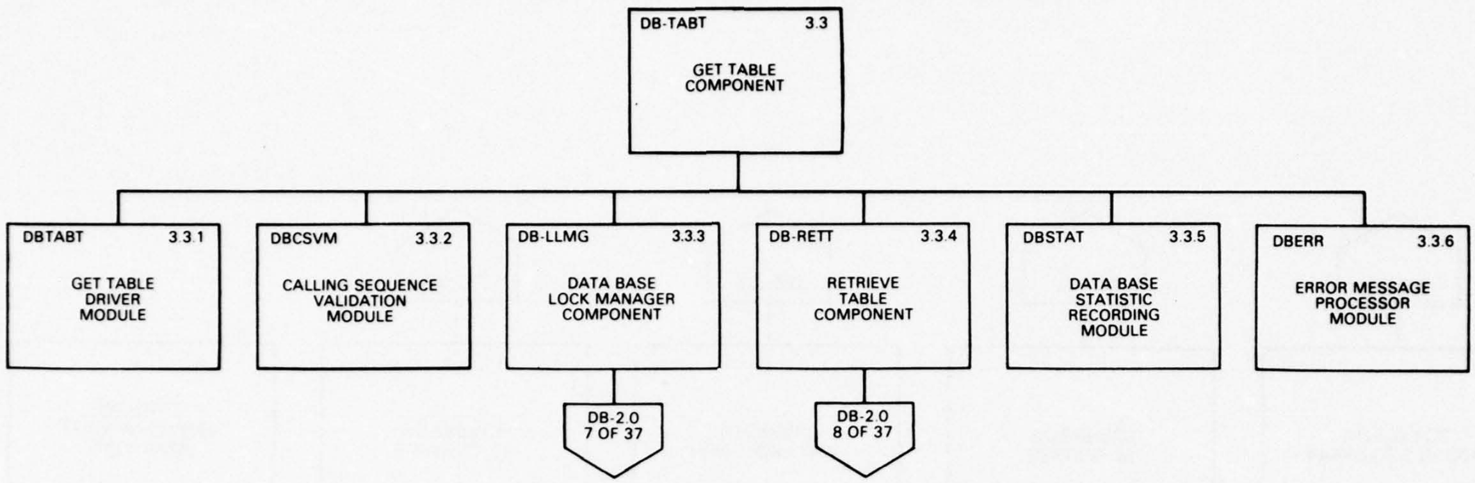


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (6 of 37)

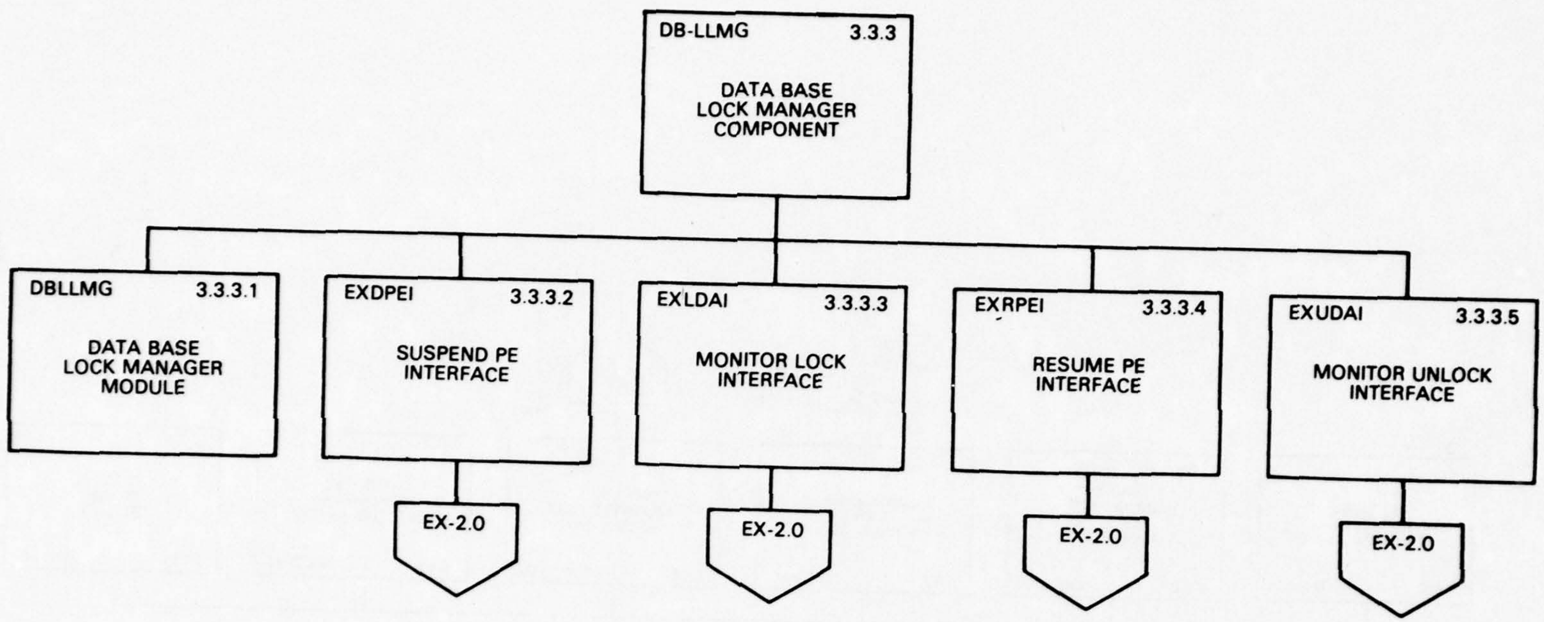


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (7 of 37)

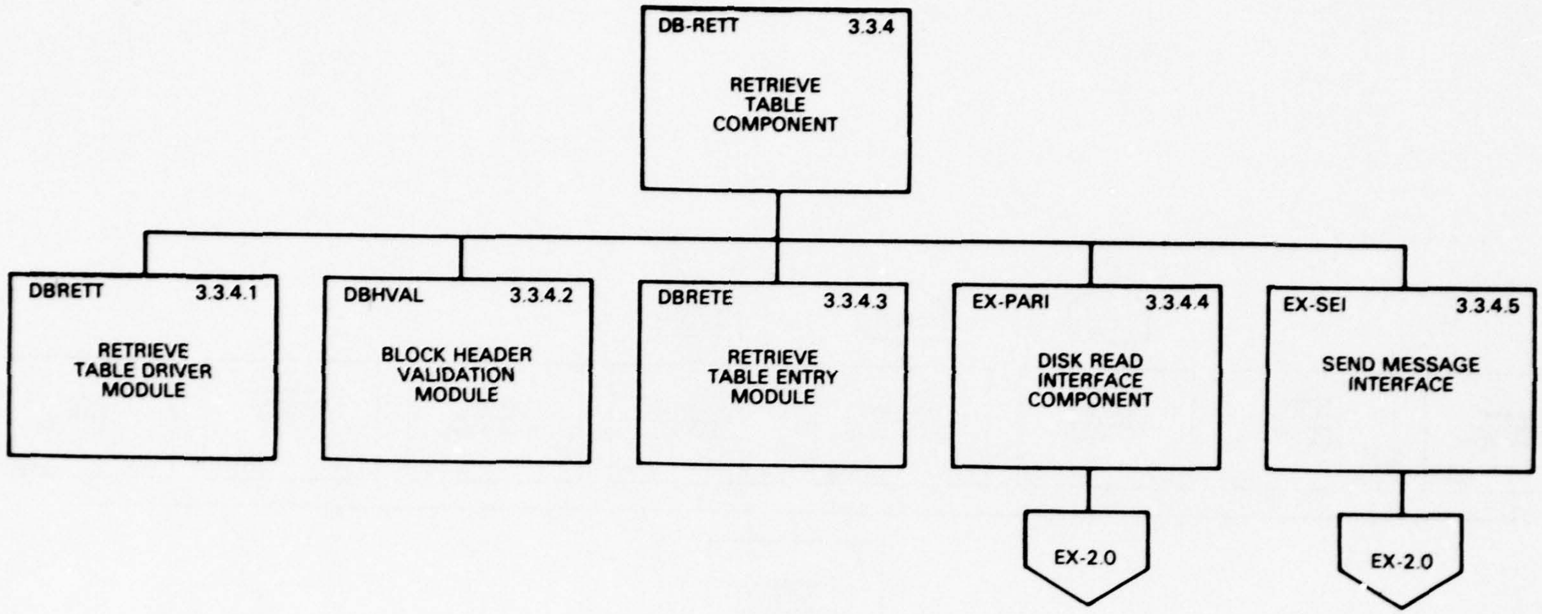


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (8 of 37)

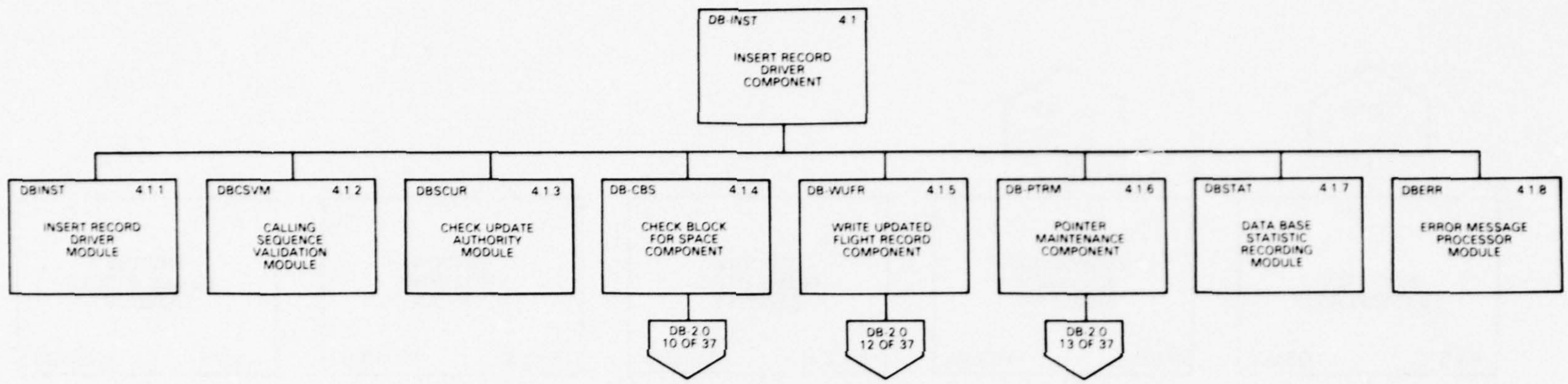


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (9 of 37)

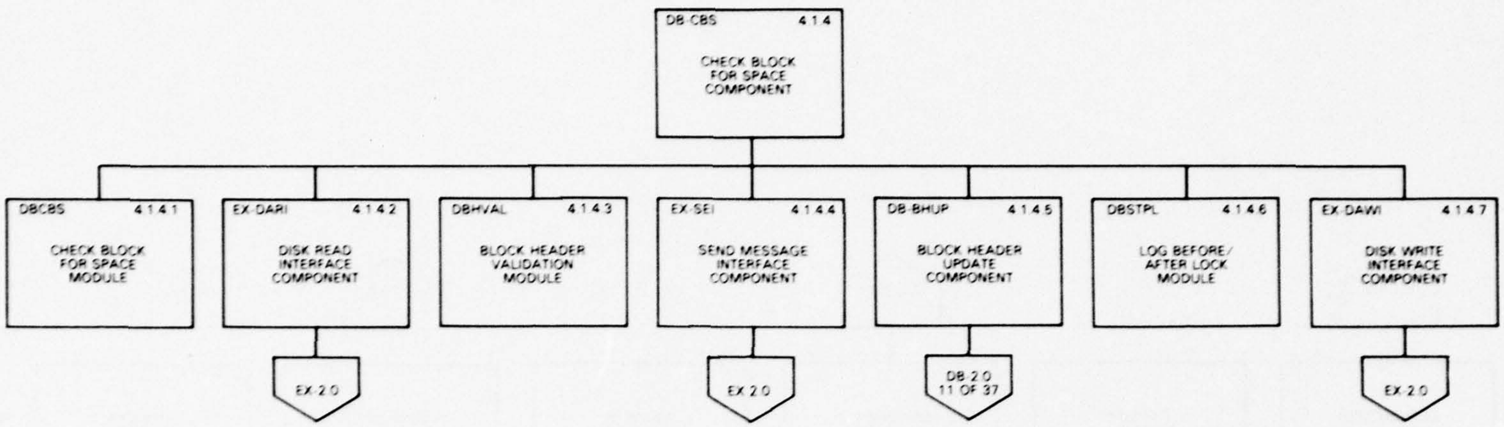


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (10 of 37)

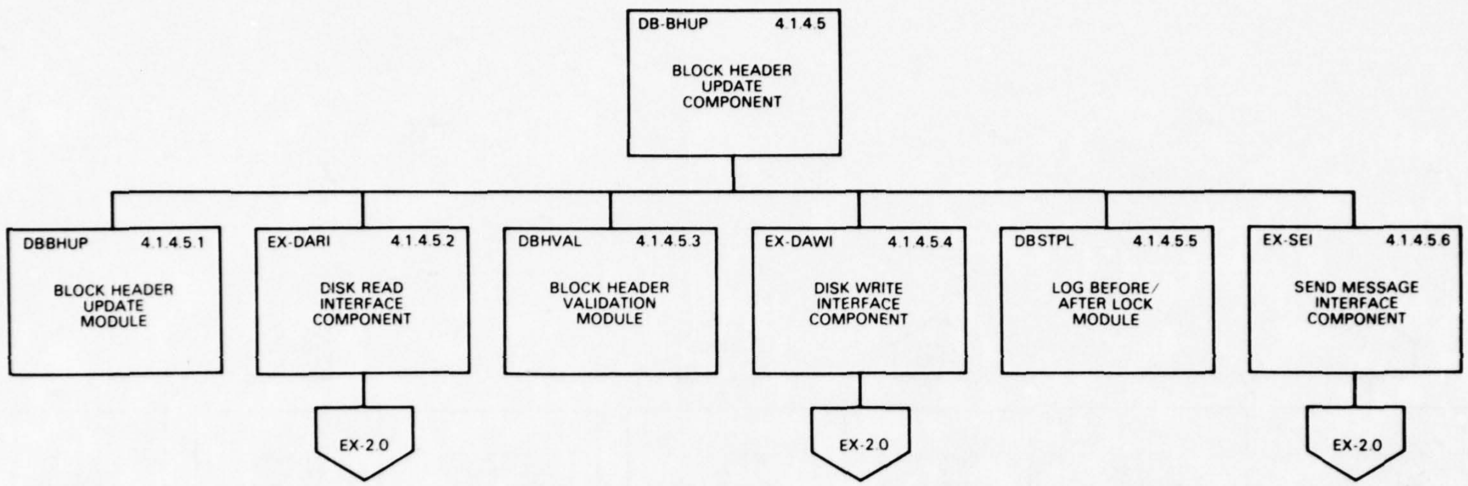


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (11 of 37)

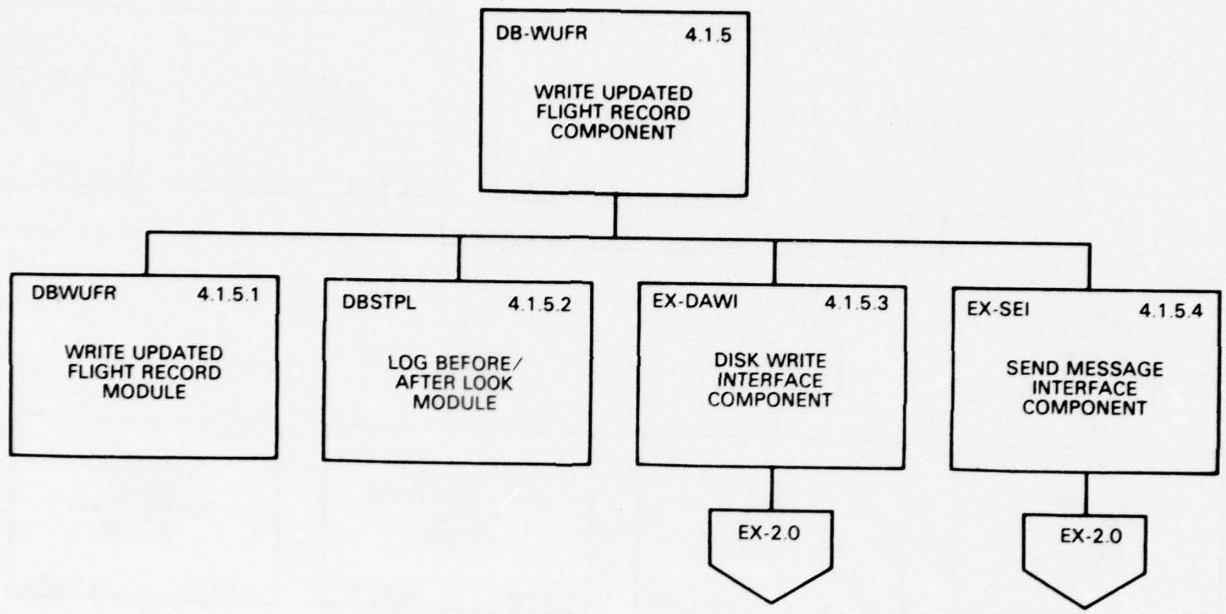


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (12 of 37)

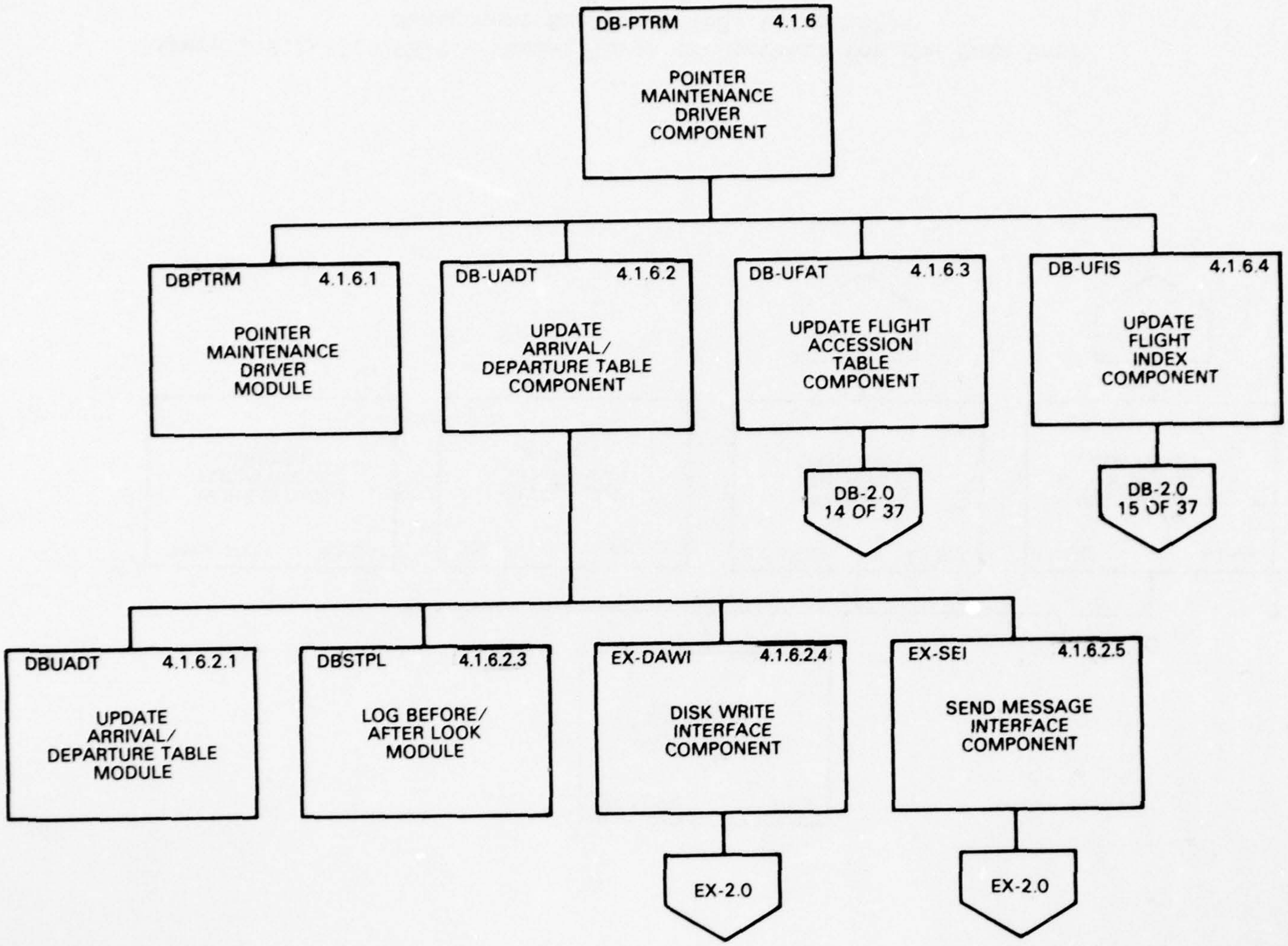


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (13 of 37)

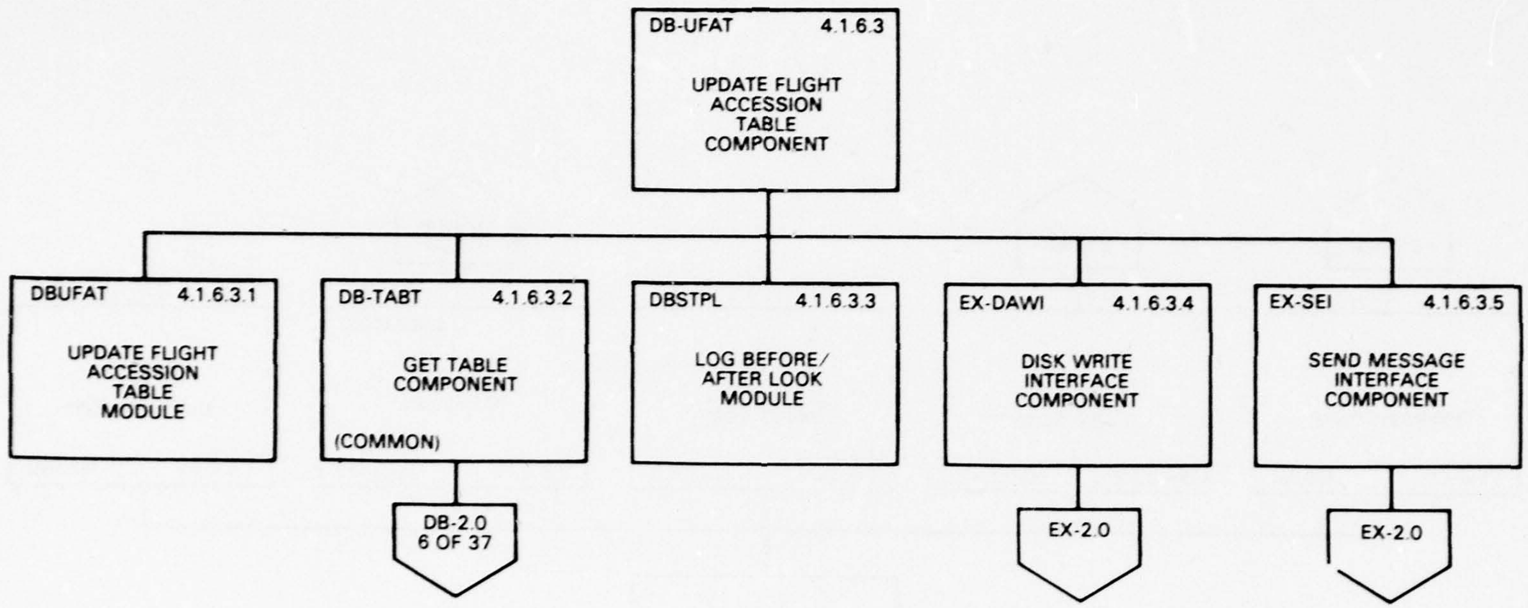


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (14 of 37)

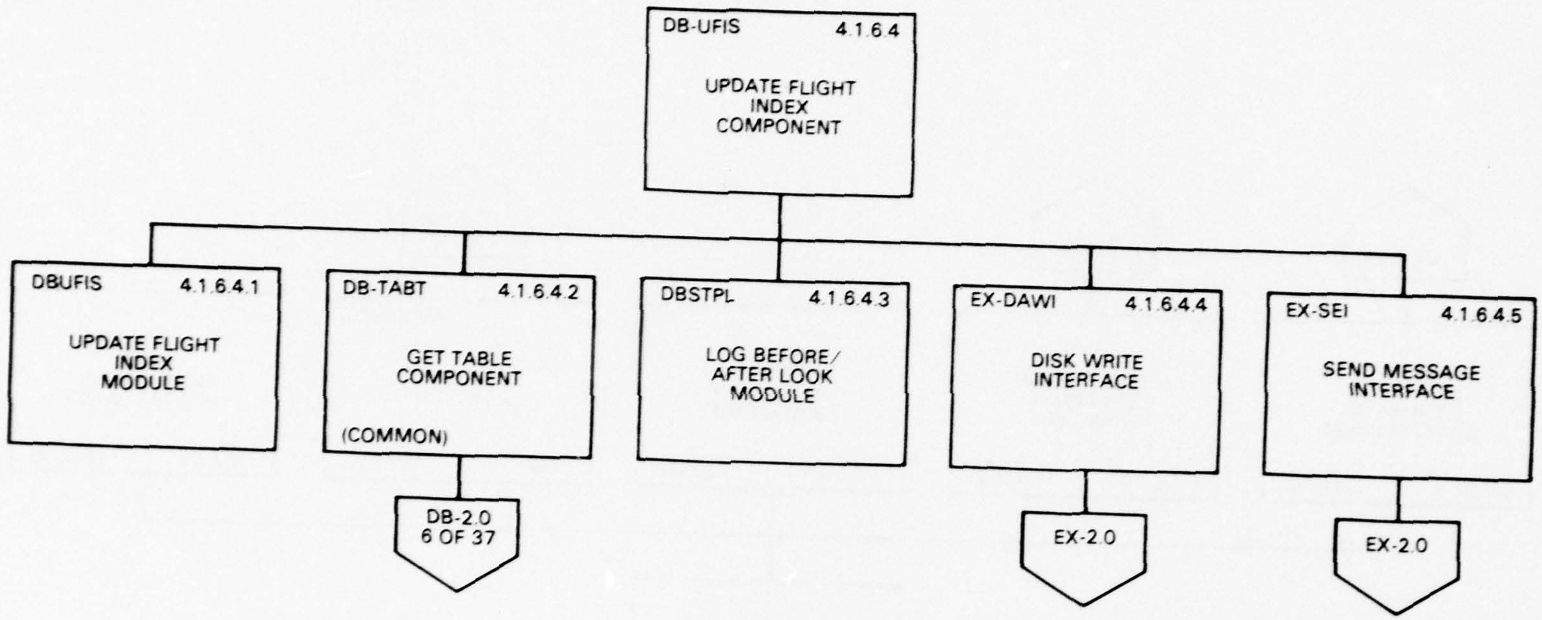


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (15 of 37)

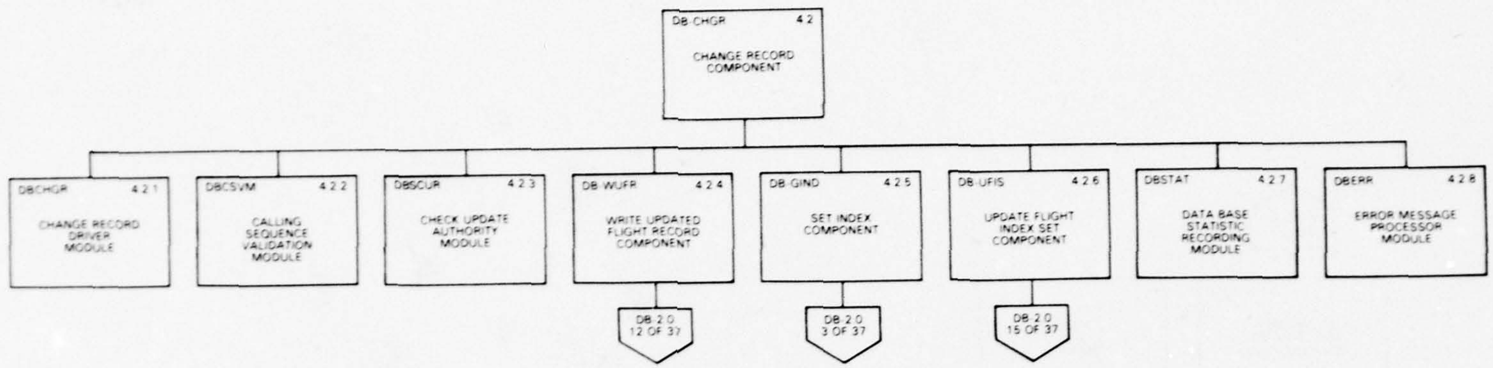


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (16 of 37)

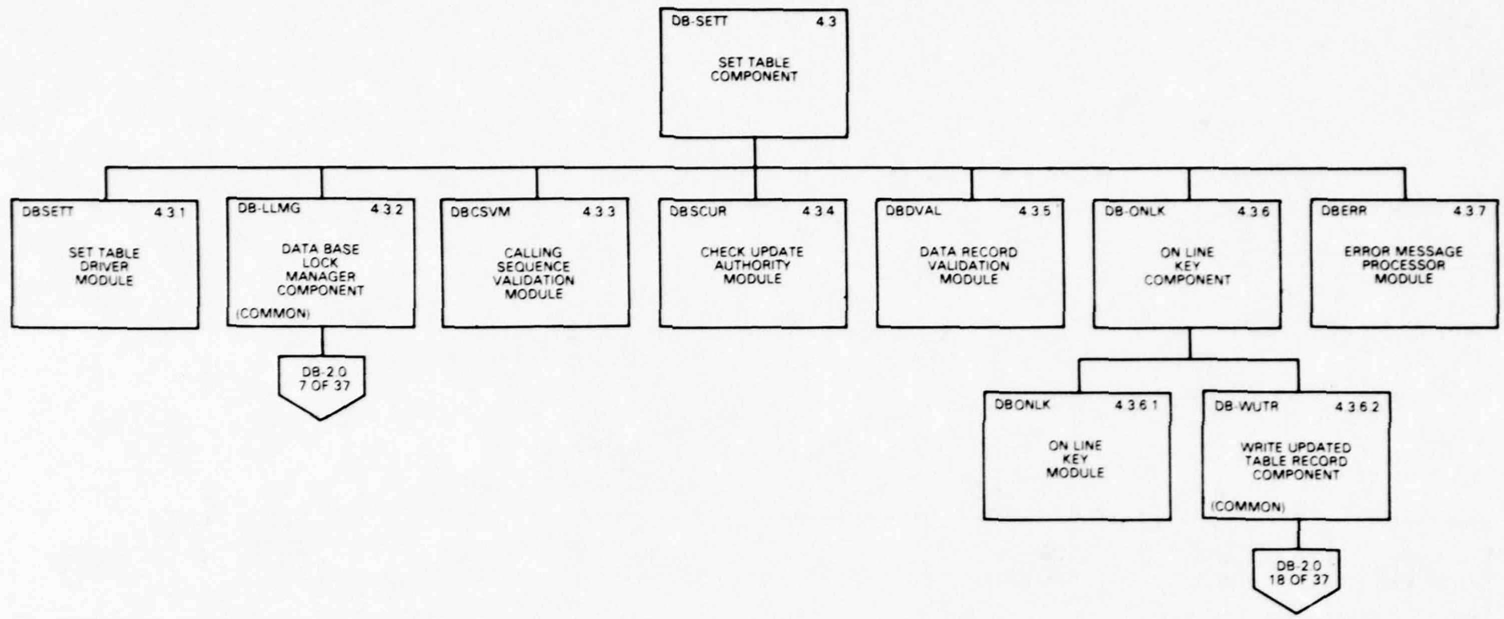


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (17 of 37)

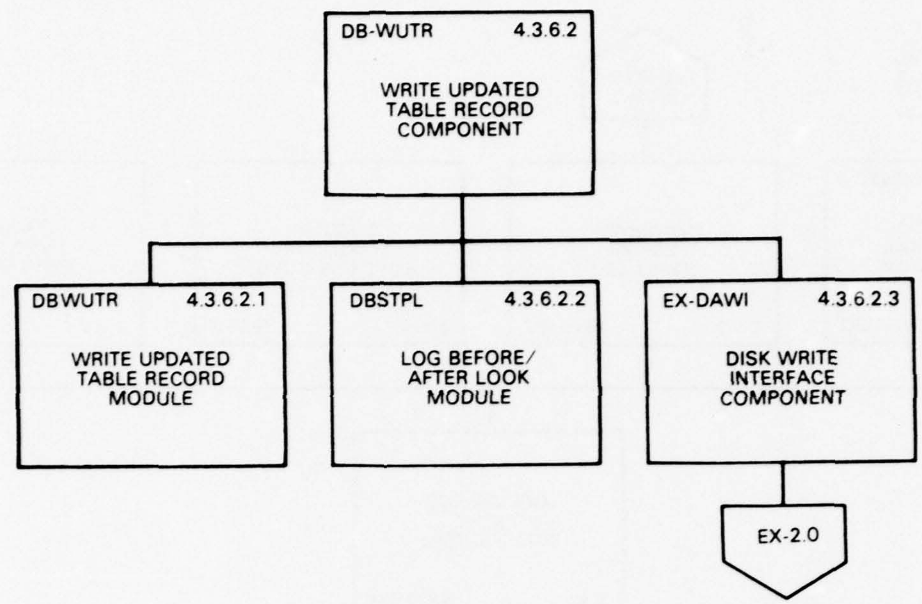


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (18 of 37)

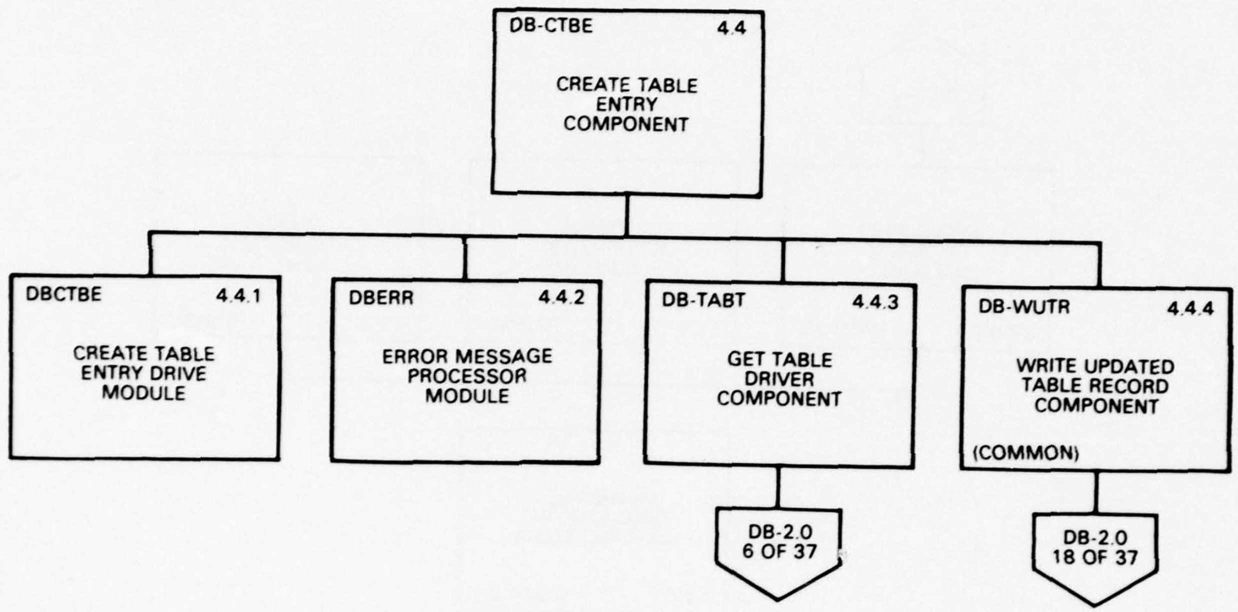


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (19 of 37)

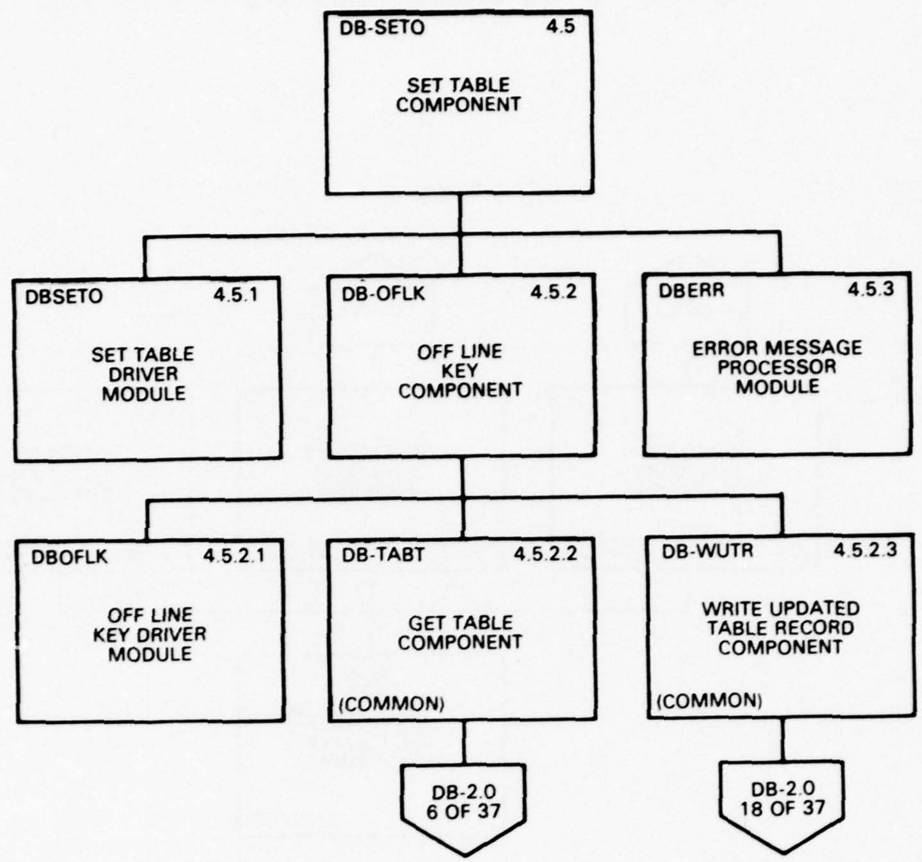


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (20 of 37)

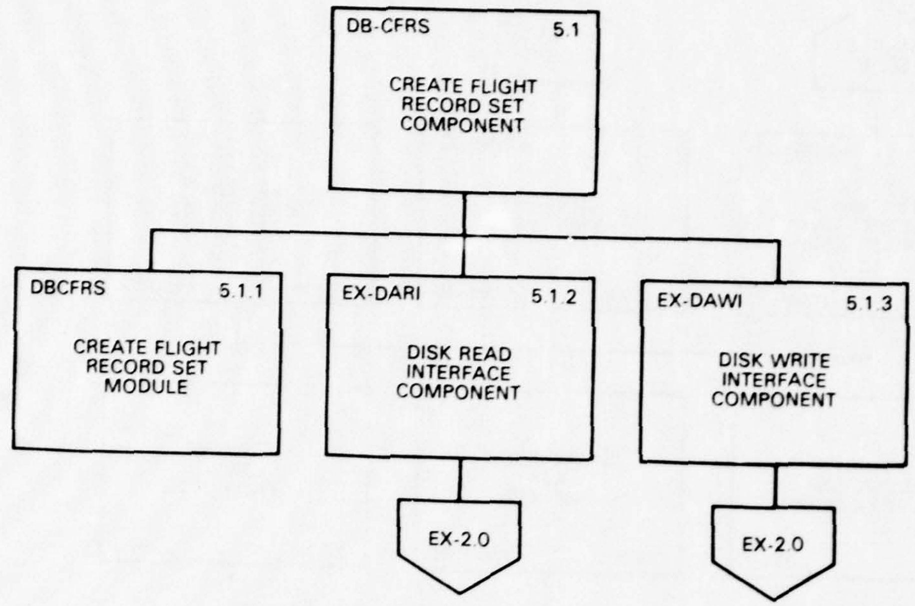


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (21 of 37)

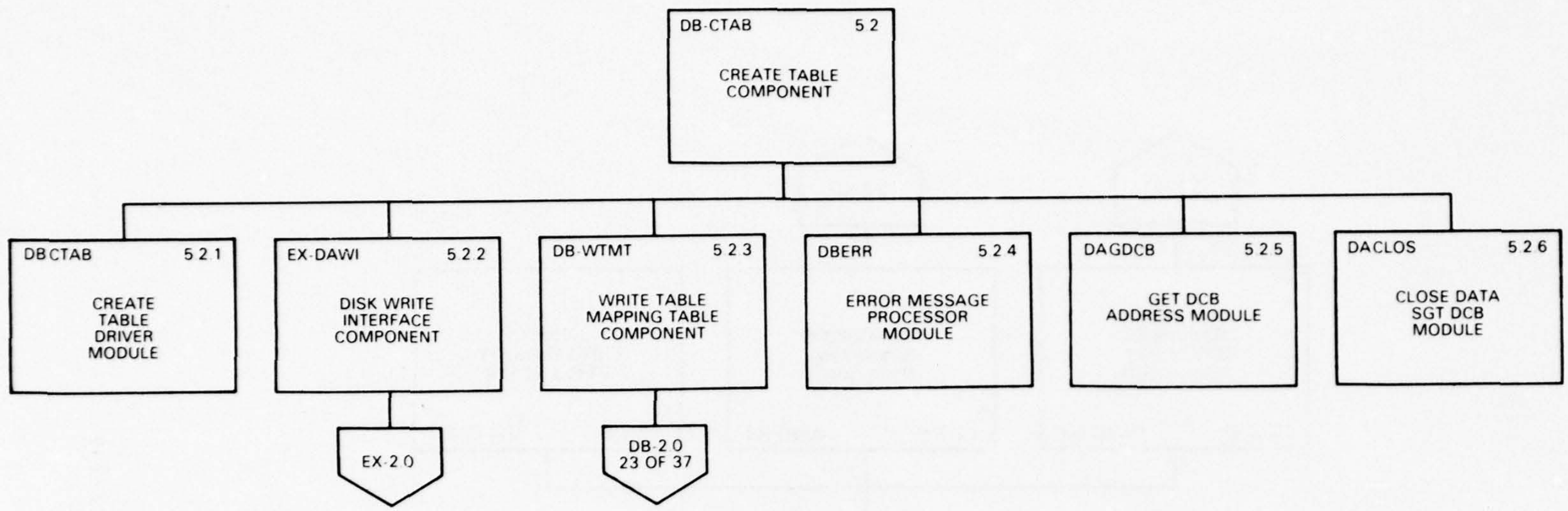


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (22 of 37)

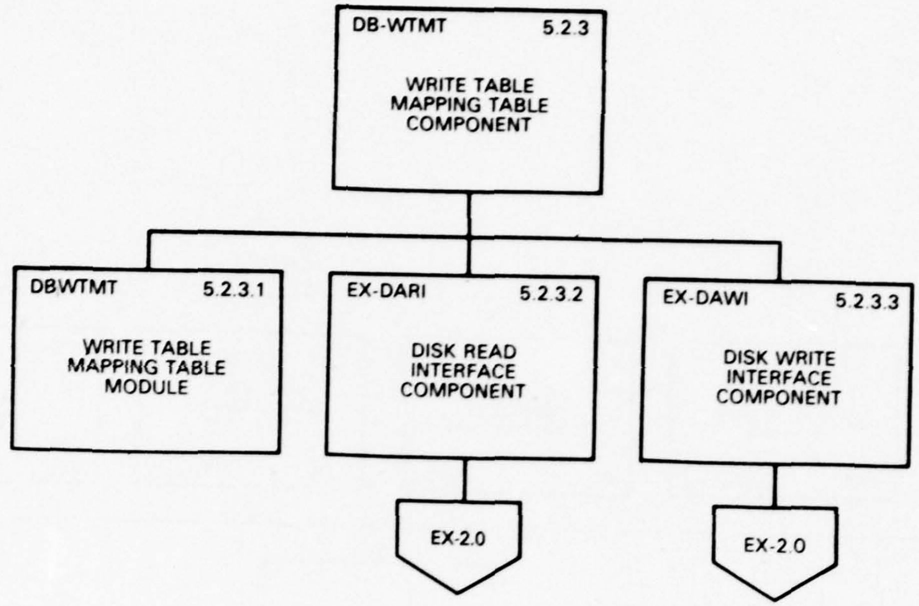


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (23 of 37)

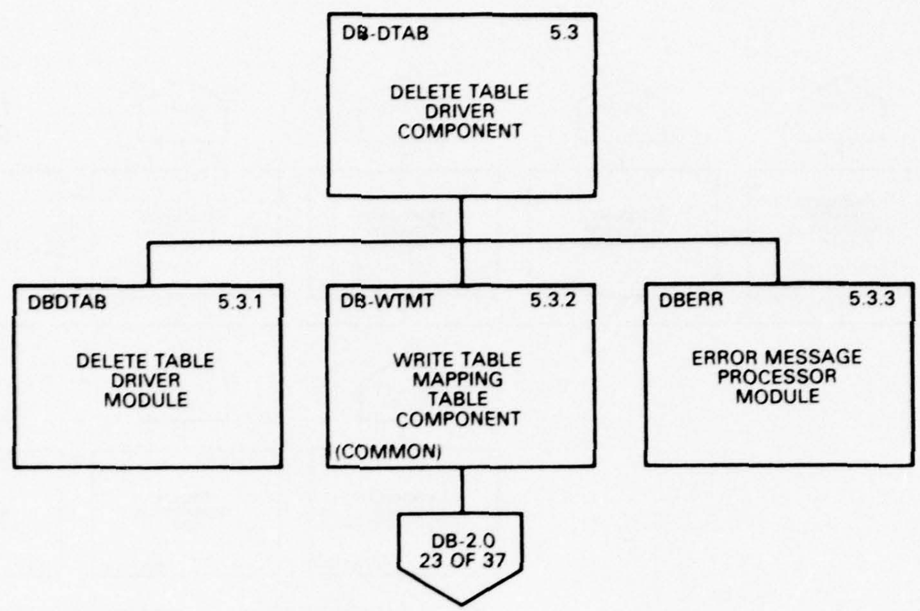


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (24 of 37)

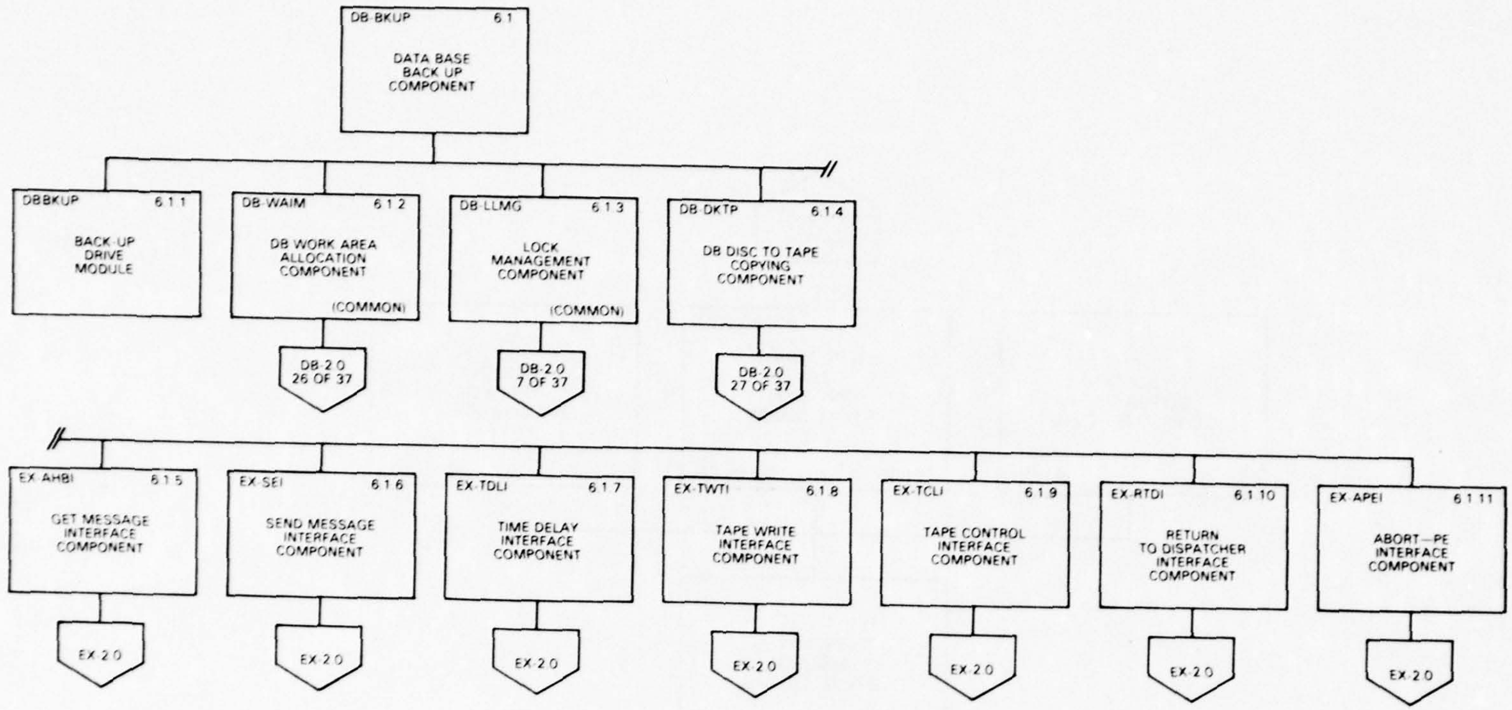


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (25 of 37)

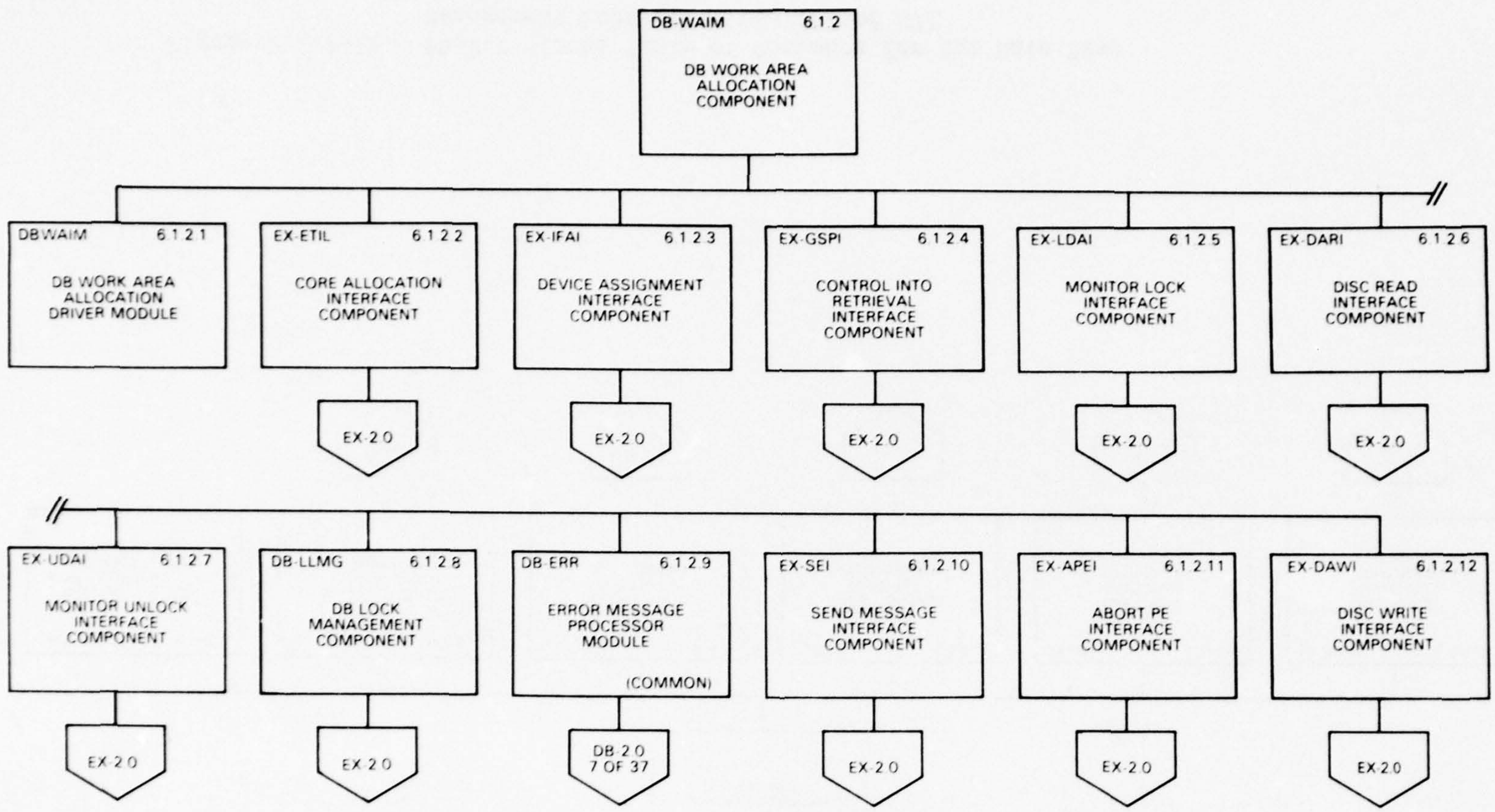


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (26 of 37)

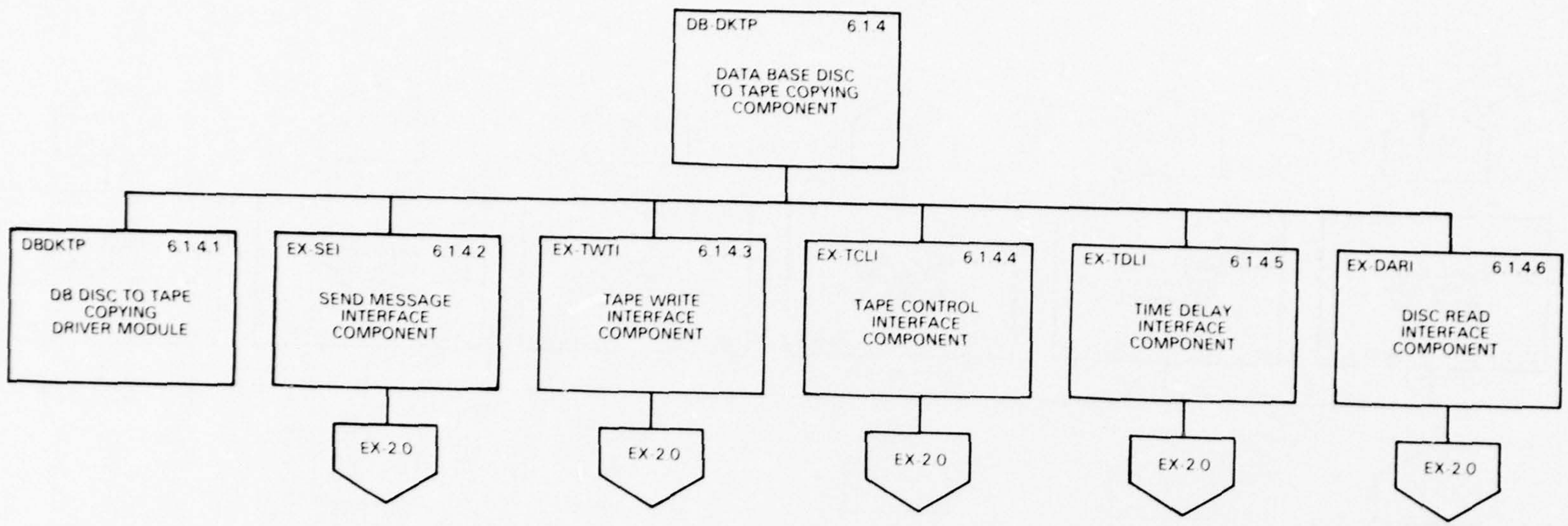


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (27 of 37)

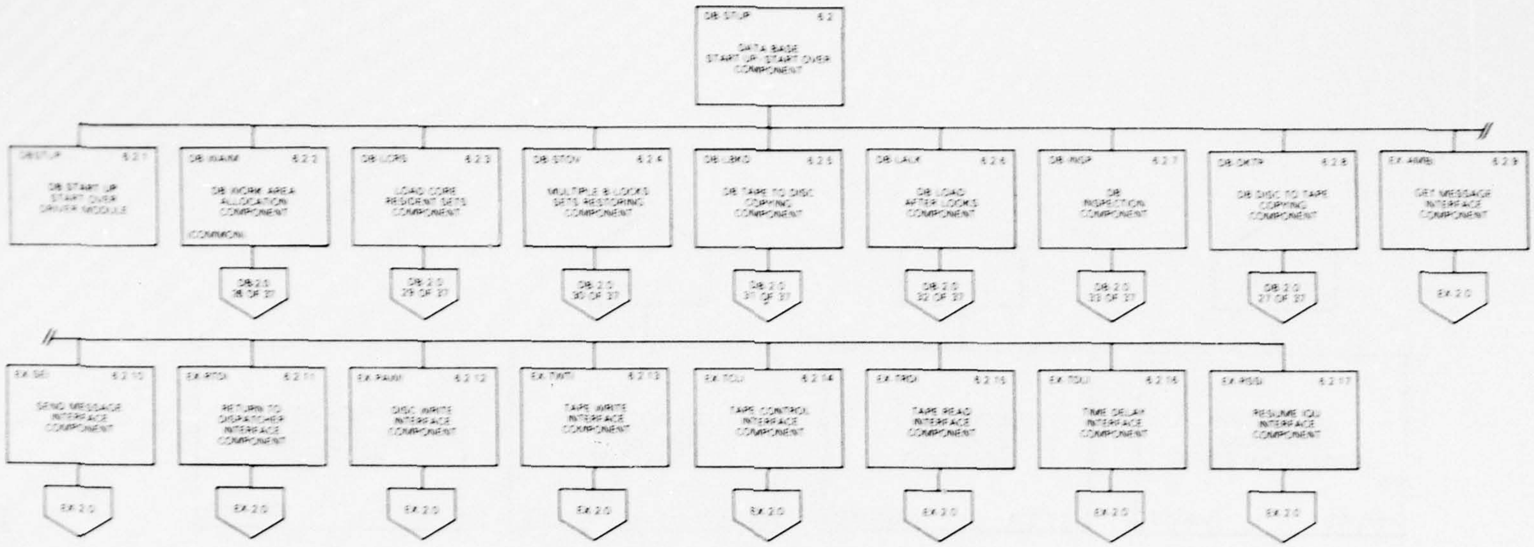


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (28 of 37)

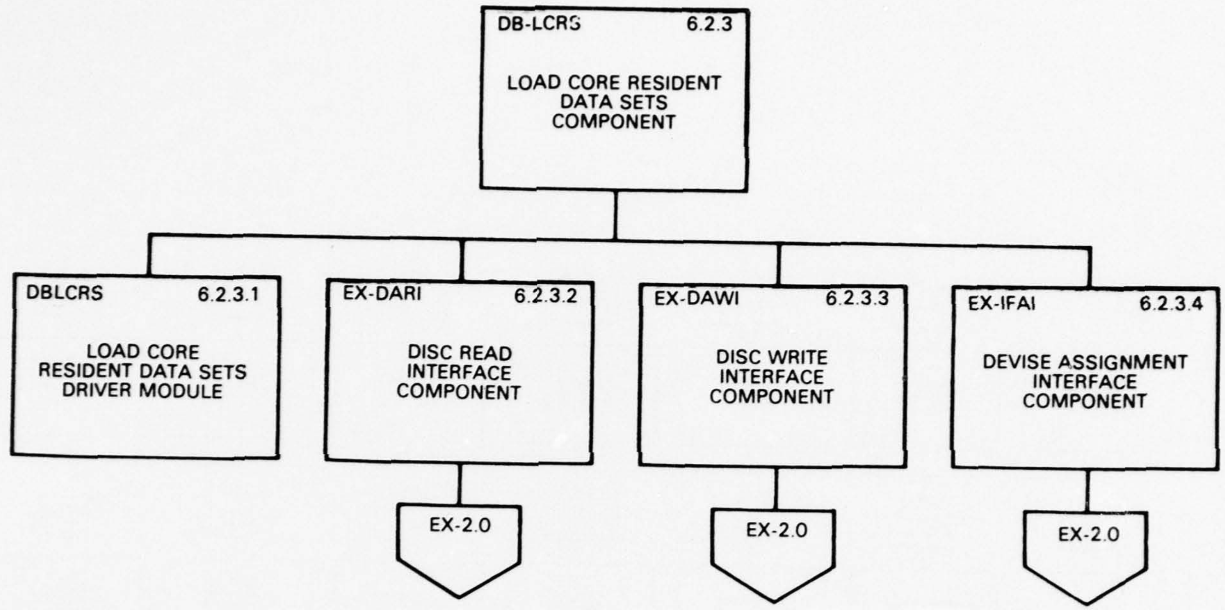


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (29 of 37)

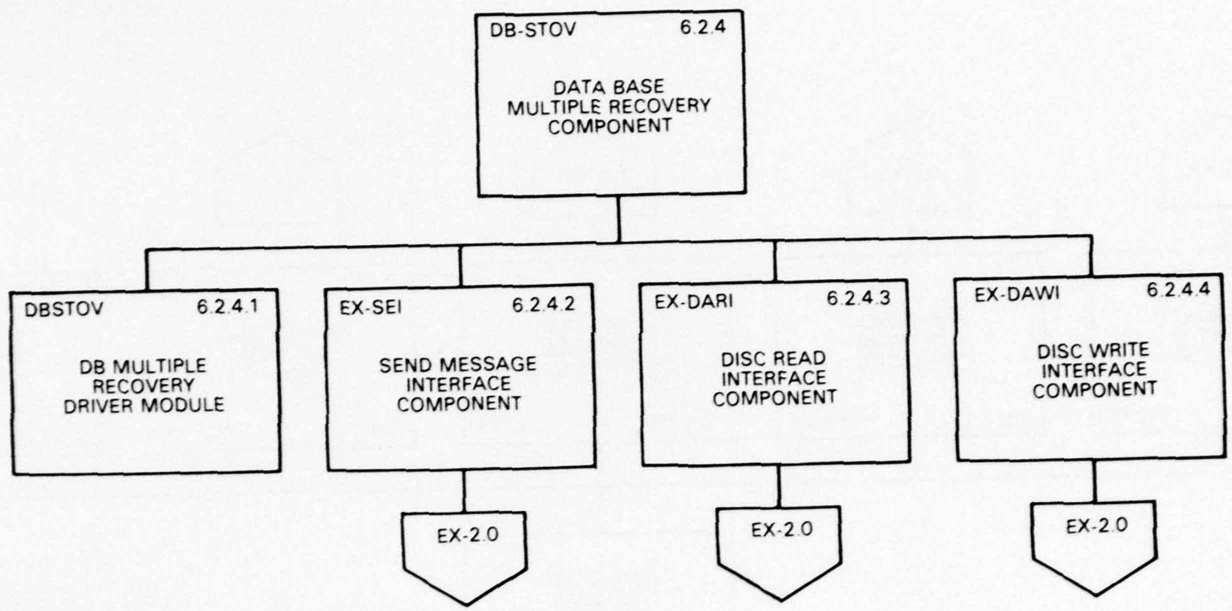


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (30 of 37)

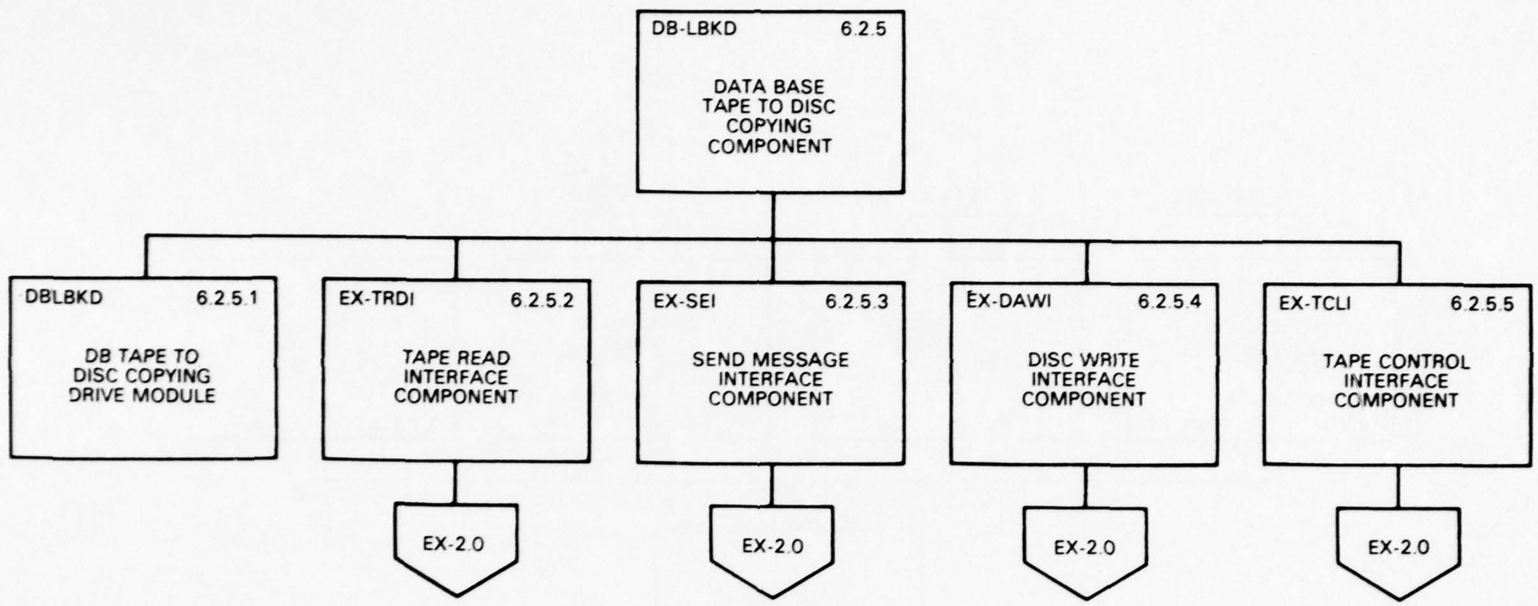


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (31 of 37)

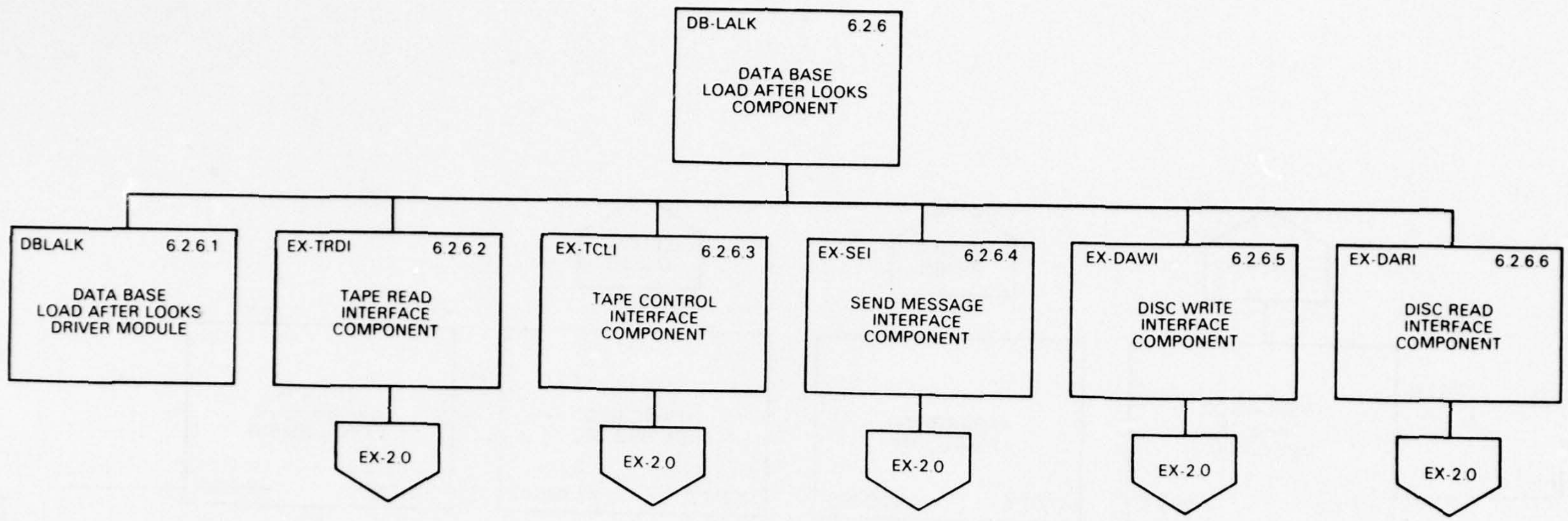


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (32 of 37)

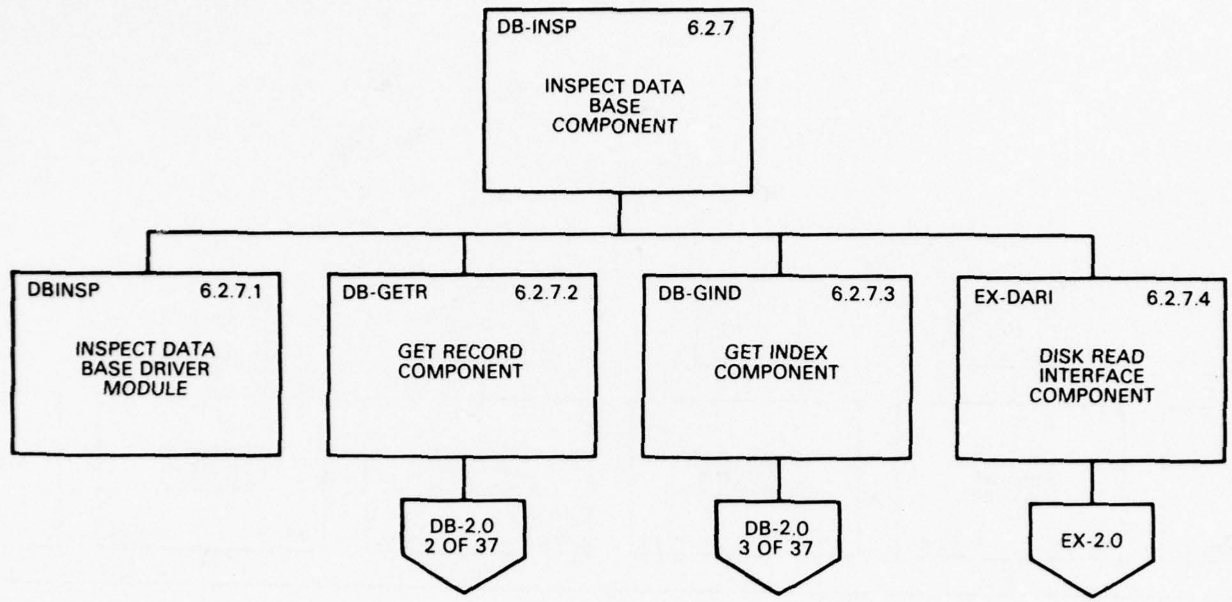


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (33 of 37)

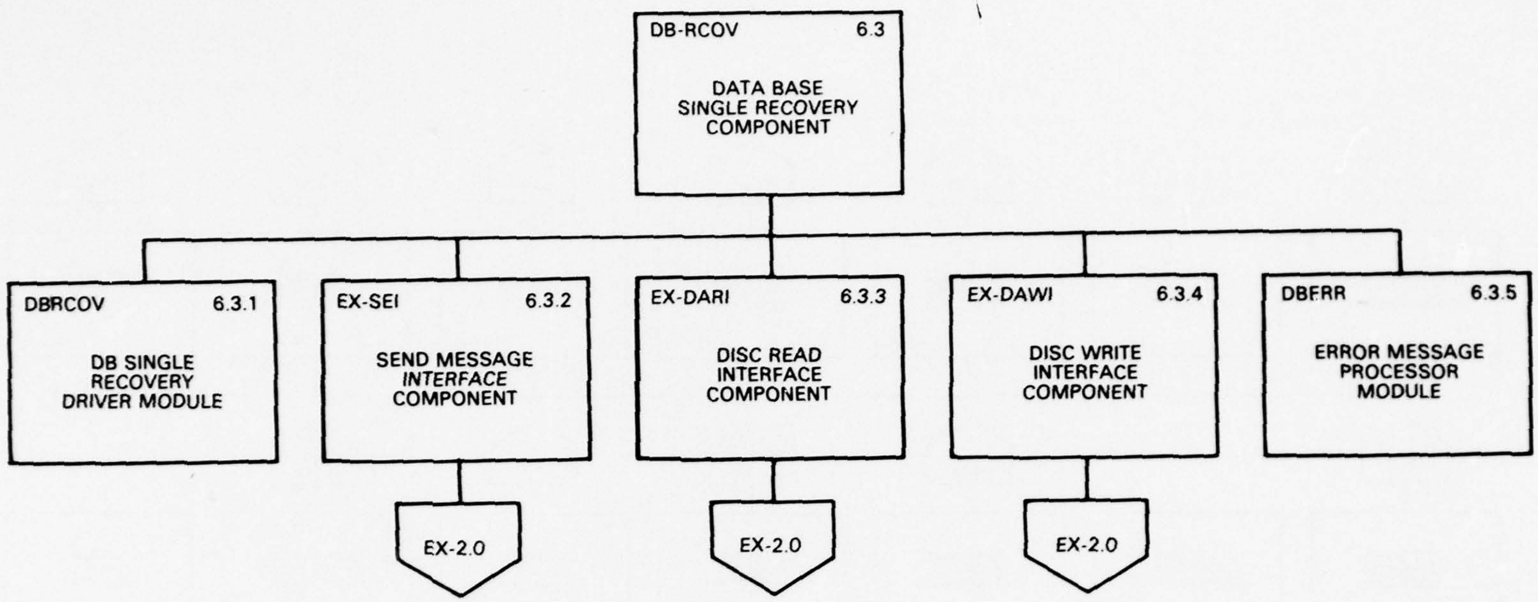


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (34 of 37)

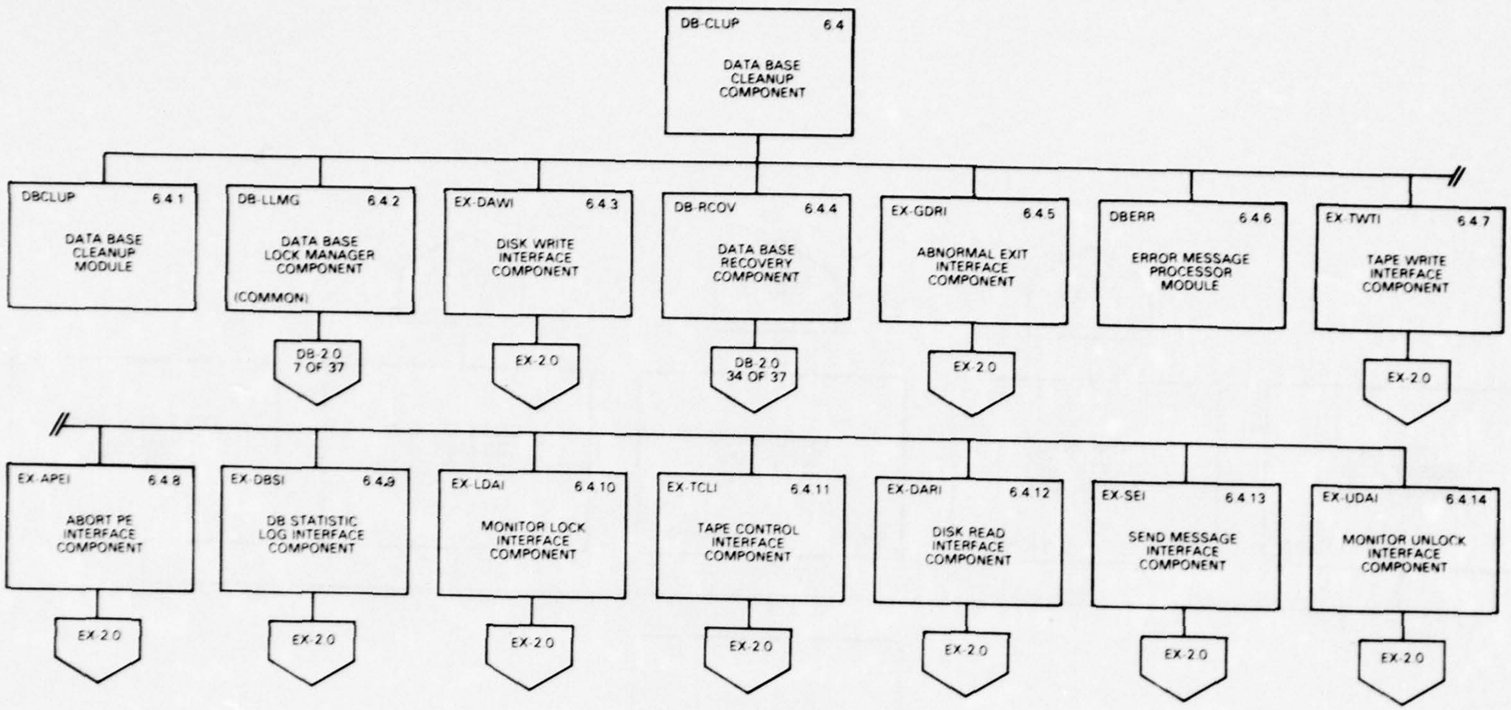


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (35 of 37)

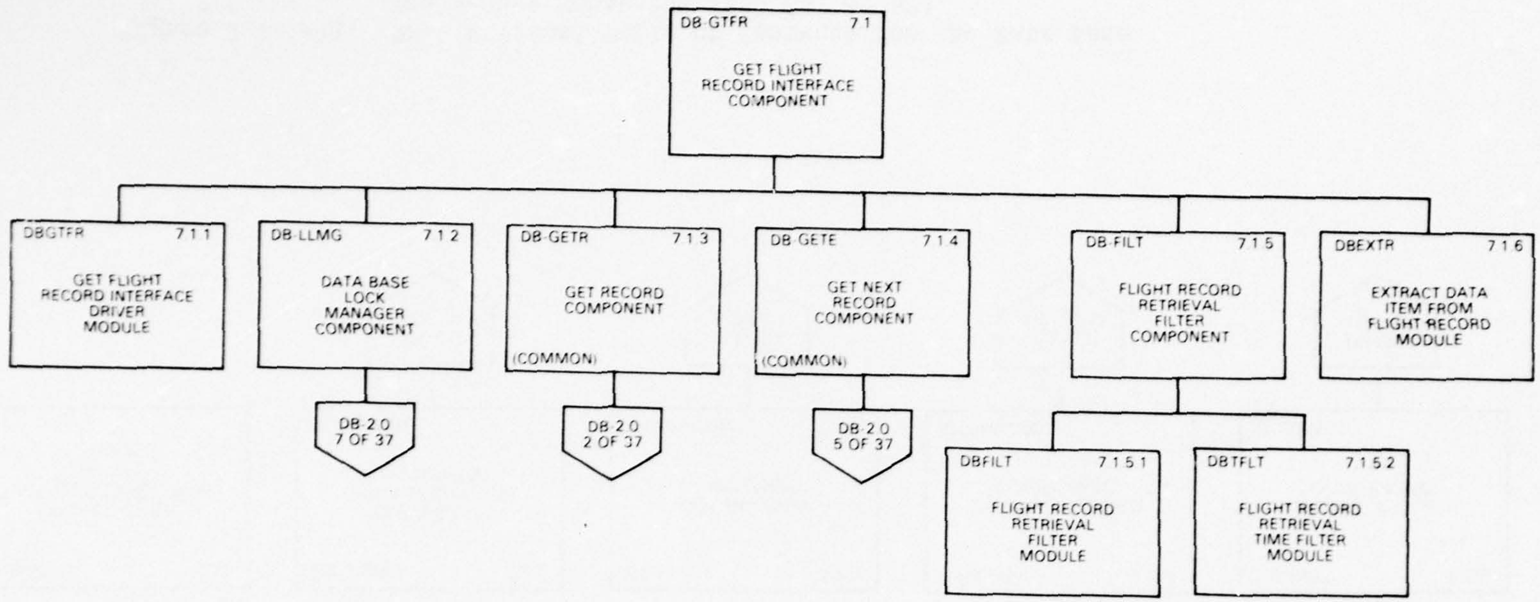


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (36 of 37)

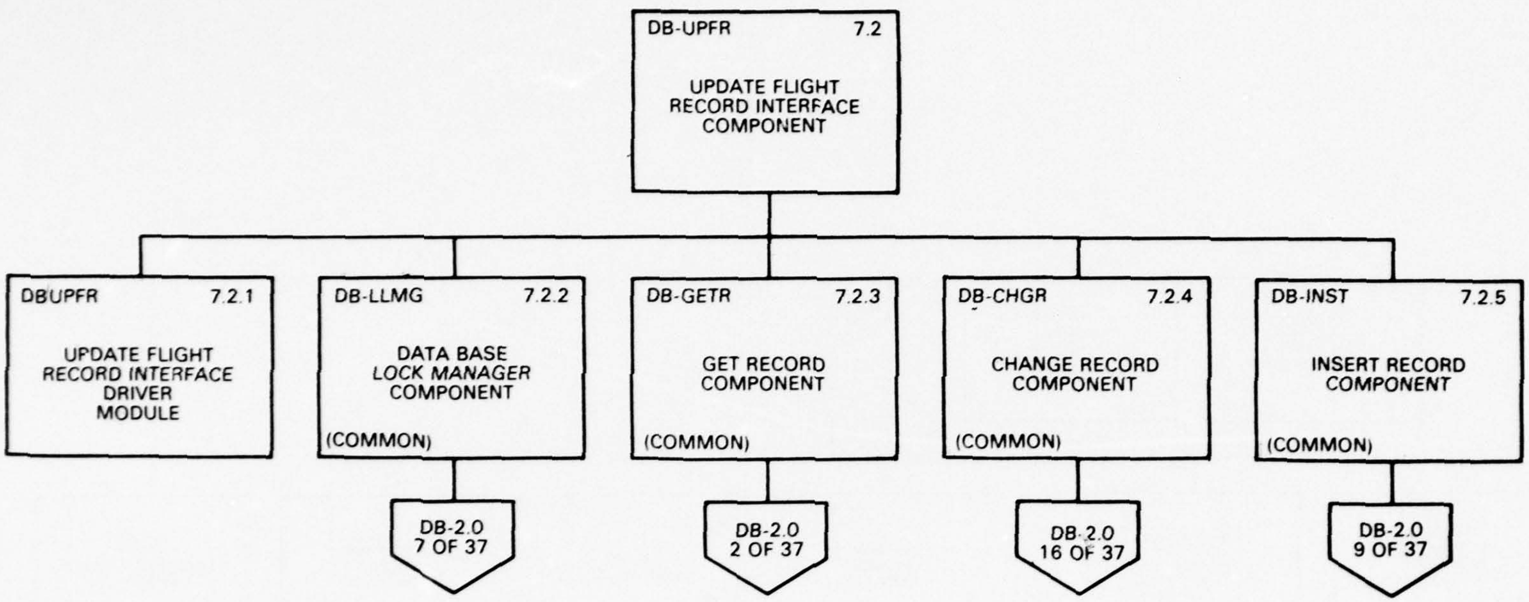


Figure 2.2.1-11. DB-2.0 Visual Table of Contents for the Data Base Management Subsystem (DB) (37 of 37)

2.2.1.3.6 Description Section for the Visual Table of Contents for the Data Base Management (DB) Subsystem

2.0 The Data Base Management Subsystem functions as the interface between all non-DB modules and the data base. It performs data base creation, inspection, update, validation, storage, retrieval, and security checking.

3.0 The DB-RETV Component retrieves flight records and table records, both primary and alternate, from the Central Flow Control Data Base Set. It consists of the DB-GETR, DB-GETE, and DB-TABT, components.

3.1 The DB-GETR Component retrieves flight records from the Central Flow Control Data Base Set. It consists of the DBGETR, DBCSVM, DBERR, DB-STAT, DB-GIND and DB-RFRB Modules and Components.

3.1.1 The DBGETR Module retrieves a flight record from the Central Flow Control Flight Record Set in accordance with the access type specified and the data provided by the caller.

3.1.2 The DBCSVM Module validates the calling sequence for the Module that requested validation.

3.1.3 The DB-GIND Component retrieves the pointer to a flight record in the Central Flow Control Flight Record Set.

3.1.3.1 The DBGIND Module retrieves the pointer to a flight record in the Central Flow Control Flight Record Set either by aircraft I.D., Airline Operator Internal Code, or Arrival/Departure Terminal.

3.1.3.2 The DB-ADT Component retrieves a pointer to a block of flight records in the flight record set.

3.1.3.2.1 The DBADT Module retrieves a pointer from the arrival/departure table and stores it in the Data Base Management Subsystem Module Parameter Set.

3.1.3.2.2 The DB-TABT Component retrieves a block of table records.

3.1.3.3 The DB-FAT Component searches for a pointer to the flight index table set based on airline operator codes.

3.1.3.3.1 The DBFAT Module stores a pointer to the flight index set in the Data Base Management Subsystem Module Parameter Set.

3.1.3.3.2 The DB-TABT Component retrieves a block of table records.

3.1.3.4 The DB-FIS Component searches for pointers to the flight record set.

3.1.3.4.1 The DBFIS Module stores a pointer to the flight record set in the Data Base Management Subsystem Module Parameter Set.

3.1.3.4.2 The DB-TABT Component retrieves a block of table records.

3.1.3.4.3 The DB-FAT Component searches for pointers to the flight index table set.

3.1.3.5 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

3.1.4 The DB-RFRB Component retrieves a flight record from the Central Flow Control Flight Record Set.

3.1.4.1 The DBRFRB Module retrieves a flight record from the Central Flow Control Flight Record Set in retrieval modes of relative record, airline operator, aircraft I.D., and arrival/departure terminal.

3.1.4.2 The EX-DARI Component performs a direct access read.

3.1.4.3 The DBHVAL Module validates all block standard headers read into core by the Data Base Management Subsystem.

3.1.4.4 The EX-SEI Component sends an error message to the operator.

3.1.4.5 The DBRFR Module scans a block of flight records for a specific flight record requested by airline operator internal code.

3.1.4.6 The EX-DAWI Component performs a direct access write.

3.1.5 The DBSTAT Module logs the access and lock statistics of the Data Base Management Subsystem.

3.1.6 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

3.2 The DB-GETE Component retrieves the next flight record with the same keys as specified by the last GETR. It consists of the DBGETE, DBERR, DB-STAT, DBRFR, and DB-GIND Modules and Components.

3.2.1 The DBGETE Module gets the next record in accordance with the access type specified and the data provided by the caller.

3.2.2 The DB-RFRB Component retrieves a flight record from the Central Flow Control Flight Record Set.

3.2.3 The DB-GIND Component retrieves the pointer to a flight record in the Central Flow Control Flight Record Set.

3.2.4 The DBSTAT Module logs the access and lock statistics of the Data Base Management Subsystem.

3.2.5 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

3.3 The DB-TABT Component retrieves table records from the Central Flow Control Data Base Set. It consists of the DBTABT, DBCSVM, DBERR, DB-STAT, DB-RETT, and DBLLMG Modules and Components.

3.3.1 The DBTABT Module retrieves a block of table records and passes back to the caller a table record in retrieval modes of key or relative record.

3.3.2 The DBCSVM Module validates the calling sequence for the caller module.

3.3.3 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

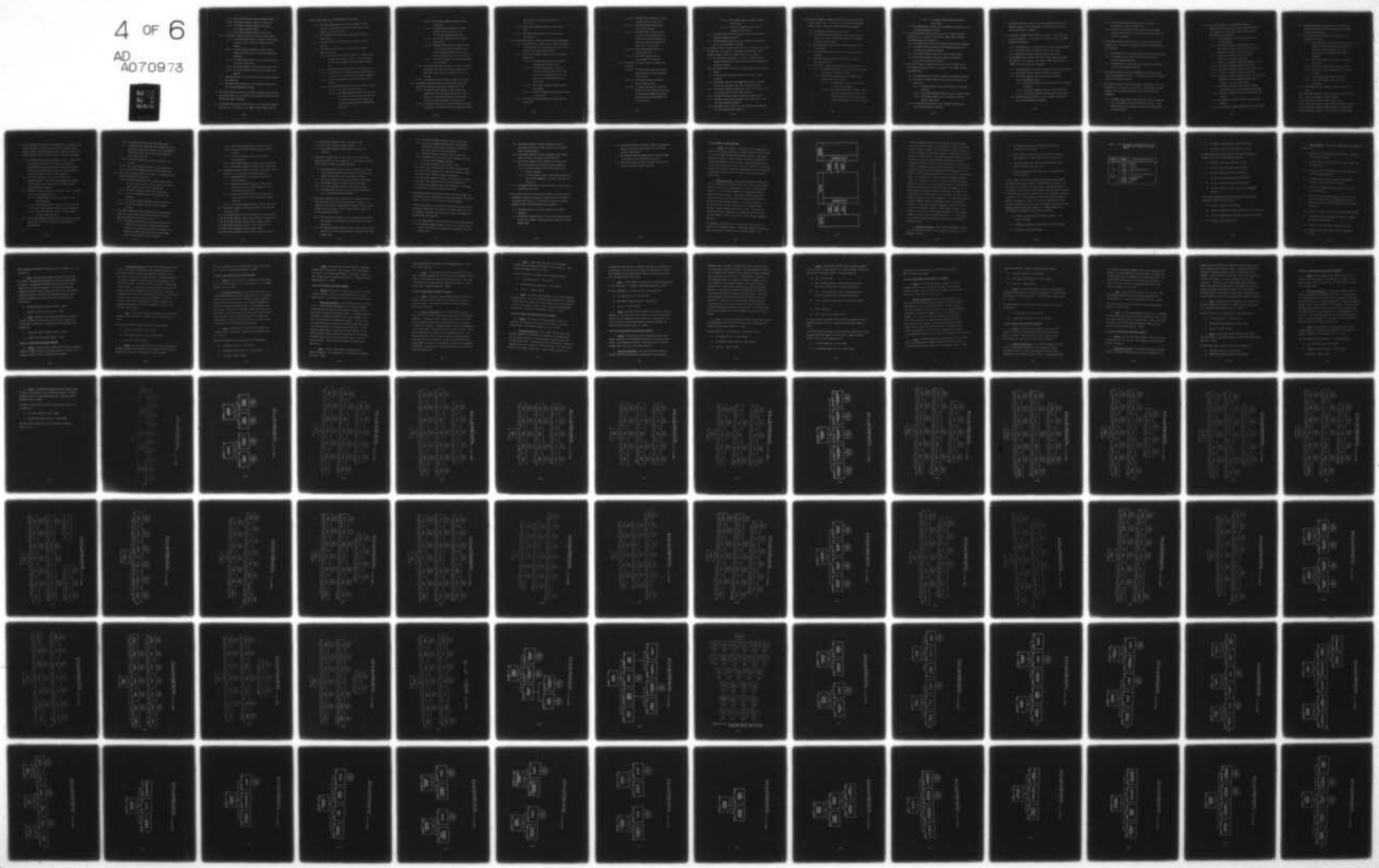
3.3.3.1 The DBLLMG Module checks validity of all lock management requests, updates the lock request queue, suspends PE's for which a lock request cannot be served and resumes PE's for which the lock request had been successfully enforced.

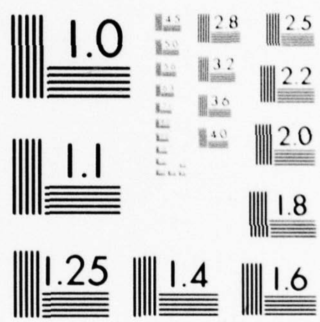
AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955

UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

4 OF 6

AD
A070973





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

3.3.3.2 The EX-DPEI Component delays a program element.

3.3.3.3 The EX-LDAI Component enqueues a data area.

3.3.3.4 The EX-RPEI Component resumes the execution of a delayed program element.

3.3.3.5 The EX-UDAI Component dequeues a data area.

3.3.4 The DB-RETT Component retrieves a block of table records.

3.3.4.1 The DBRETT Module retrieves a block of table records and makes a call to retrieve a specified table record.

3.3.4.2 The DBHVAL Module validates all block standard headers read into core by the Data Base Management Subsystem.

3.3.4.3 The DBRETE Module scans a block of table records for a specified table record.

3.3.4.4 The EX-DARI Component performs a direct access read.

3.3.4.5 The EX-SEI Component sends an error message to the operator.

3.3.5 The DBSTAT Module logs the statistics of the Data Base Management Subsystem.

3.3.6 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

4.0 The DB-UPD Component initiates and controls the update processing for all data base data sets. It consists of the DB-INST, DB-GHGR, DB-SETT, DB-CTBE and DB-SETO Components.

4.1 The DB-INST Components add flight records to the Central Flow Control Data Base Set. It consists of the DBINST, DBCSVM, DBSCUR, DB-STAT,

DB-CBS, DBERR, DBWUFR, and DB-PTRM Modules and Components.

4.1.1 The DBINST Module makes all the calls to insert a flight record into the Central Flow Control Flight Record Set and updates the associated indicies.

4.1.2 The DBCSVM Module validates the calling sequence for the caller module.

4.1.3 The DBSCUR Module checks the monitor bit mask for update authority.

4.1.4 The DB-CBS Component checks for free space in a flight record set block.

4.1.4.1 The DBCBS Module finds free space in a flight record set block and stores a pointer to that free space in the Data Base Management Subsystem Module Parameter Set.

4.1.4.2 The EX-DARI Component performs a direct access read.

4.1.4.3 The DBHVAL Module validates all block standard headers read into core by the Data Base Management Subsystem.

4.1.4.4 The EX-SEI Component sends an error message to the operator.

4.1.4.5 The DB-BHUP Component retrieves blocks from the flight record set and the flight index table set.

4.1.4.5.1 The DBBHUP Module retrieves prime and overflow blocks from the flight record set and the flight index table set, updates the block header and updates the housekeeping table set.

4.1.4.5.2 The EX-DARI Component performs a direct access read.

4.1.4.5.3 The DBHVAL Module validates all block standard headers read into core by the Data Base Management Subsystem.

4.1.4.5.4 The EX-DAWI Component performs a direct access write.

4.1.4.5.5 The DBSTPL Module builds a before or after look record according to an index in the caller sequences, and writes the record into the before/after look set for the TLM.

4.1.4.5.6 The EX-SEI Component sends an error message to the operator.

4.1.4.6 The DBSTPL Module builds a before or after look record according to an index in the caller sequences, and writes the record into the before/after look set for the TLM.

4.1.4.7 The EX-DAWI Component performs a direct access write.

4.1.5 The DB-WUFR Component inserts one updated and/or new flight record into the Central Flow Control Flight Record Set.

4.1.5.1 The DBWUFR Module inserts one updated and/or new flight record into the Central Flow Control Flight Record Set, calculates the relative record number in the set and stores the updated record on disk.

4.1.5.2 The DBSTPL Module builds a before or after look record according to an index in the caller sequence and writes

the record into the before/after look set for the TLM.

4.1.5.3 The EX-DAWI Component performs a direct access write.

4.1.5.4 The EX-SEI Component sends an error message to the operator.

4.1.6 The DB-PTRM Component performs all pointer record maintenance.

4.1.6.1 The DBPTRM Module makes calls to update the arrival/departure table (ADT), flight accession table (FAT), and the flight index table set (FIS).

4.1.6.2 The DB-UADT Component updates the arrival/departure table (ADT).

4.1.6.2.1 The DBUADT Module updates the arrival/departure table internal codes.

4.1.6.2.2 The DBSTPL Module builds a before or after look record according to an index in the caller sequences, and writes the record into the before/after look set for the TLM.

4.1.6.2.3 The EX-DAWI Component performs a direct access write.

4.1.6.2.4 The EX-SEI Component sends an error message to the operator.

4.1.6.3 The DB-UFAT Component updates the flight accession table (FAT).

4.1.6.3.1 The DBUFAT Module updates the flight accession table by aircraft I.D.

4.1.6.3.2 The DB-TABT Component retrieves a block of table records.

4.1.6.3.3 The DBSTPL Module builds a before or after look record according to an index in the calling sequences, and writes the record into the before/after look set for the TLM.

4.1.6.3.4 The EX-DAWI Component performs a direct access write.

4.1.6.3.5 The EX-SEI Component sends an error message to the operator.

4.1.6.4 The DB-UFIS Component updates the flight index set (FIS).

4.1.6.4.1 The DBUFIS Module updates the flight index set using the airline operator internal code.

4.1.6.4.2 The DB-TABT Component retrieves a block of table records.

4.1.6.4.3 The DBSTPL Module builds a before or after look record according to an index in the calling sequences, and writes the record into the before/after look set for the TLM.

4.1.6.4.4 The EX-DAWI Component performs a direct access write.

4.1.6.4.5 The EX-SEI Component sends an error message to the operator.

4.1.7 The DBSTAT Module logs the access and lock statistics of the Data Base Management Subsystem.

4.1.8 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

4.2 The DB-CHGR Component changes flight records of the Central Flow Control Data Base Set. It consists of the DBCHGR, DBCSVM, DBERR, DBSCUR, DB-STAT, DB-GEND, DB-UFES, and DBWUFR Modules and Components.

4.2.1 The DBCHGR Module determines if this is a record requiring changes to data items or a new record and calls the appropriate lower level modules.

4.2.2 The DBCSVM Module validates the calling sequence for the caller module.

4.2.3 The DBSCUR Module checks the monitor bit mask for update authority.

4.2.4 The DB-WUFR Component inserts one updated and/or new flight record into the Central Flow Control Flight Record Set.

4.2.5 The DB-GIND Component retrieves the pointer to a flight record in the Central Flow Control Flight Record Set.

4.2.6 The DB-UFIS Component updates the flight index set (FIS).

4.2.7 The DBSTAT Module logs the access and lock statistics of the Data Base Management Subsystem.

4.2.8 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

4.3 The DB-SETT Component changes table records of the Central Flow Control Data Base Set. It consists of the DBSETT, DBCSVM, DBERR, DBSCUR, DB-STAT, DB-LLMG, and DB-ONLK Modules and Components.

4.3.1 The DBSETT Module updates a table record.

4.3.2 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

4.3.3 The DBCSVM Module validates the calling sequence for the caller module.

4.3.4 The DBSCUR Module checks the monitor bit mask for update authority.

4.3.5 The DBDVAL Module validates the input data to update a table record on-line.

4.3.6 The DB-ONLK Component checks validity of an access key.

4.3.6.1 The DBONLK Module makes the call to write the updated table record if the key is valid.

4.3.6.2 The DB-WUTR Component writes the updated table record on the CFC data base.

4.3.6.2.1 The DBWUTR Module is the driver module for the DB-WUTR Component.

4.3.6.2.2 The DBSTPL Module builds a before or after look record according to an index in the calling sequences, and writes the record into the before/after look set for the TLM.

4.3.6.2.3 The EX-DAWI Component performs a direct access write.

4.3.7 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

4.4 The DB-CTBE Component adds table records to the Central Flow Control Data Base Set. It consists of the DBCTBE, DBERR, DBDVAL, DB-TABT, and DBWUTR Modules and Components.

4.4.1 The DBCTBE Module is the driver module for the DB-CTBE Component.

4.4.2 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

4.4.3 The DB-TABT Component retrieves a block of table records.

4.4.4 The DB-WUTR Component writes the updated table record on the Central Flow Control data base.

4.5 The DB-SETO Component changes table records of the CFC data base for the off-line data base functions. It consists of the DBSETO, DB-OFLK, and DBERR Modules.

4.5.1 The DBSETO Module is the driver module for the DB-SETO Component.

4.5.2 The DB-OFLK Component checks validity of an off-line data base access key.

4.5.2.1 The DBOFLK Module is the driver module for the DB-OFLK Component.

4.5.2.2 The DB-TABT Component retrieves a block of table records.

4.5.2.3 The DB-WUTR Component is the interface to write the updated table record.

4.5.3 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

5.0 The DB-INIT Component creates the data base flight record files and data base tables in the off-line mode. It consists of the DB-GFRS, DB-CTAB, and DB-DTAB Components.

5.1 The DB-CFRS Component creates flight record sets of the Central Flow Control Data Base Set. It consists of the DBCFRS, EXDARI, and EXDAWF Modules and Components.

5.1.1 The DBCFRS Module is the driver module for the DB-CFRS Component.

5.1.2 The EX-DARI Component performs a direct access read.

5.1.3 The EX-DAWI Component performs a direct access write.

5.2 The DB-CTAB Component creates table sets of the Central Flow Control Data Base Set. It consists of the DBCTAB, EXDAWC, DBERR, DAG-DCB, DBWTMT, and DACLOS Modules and Components.

5.2.1 The DBCTAB Module is the driver module for the DB-CTAB Component.

5.2.2 The EX-DAWI Component performs a direct access write.

5.2.3 The DB-WTMT Component inserts and/or deletes entries in the table mapping table set.

5.2.3.1 The DB-WTMT Module is the driver module for the DB-WTMT Component.

5.2.3.2 The EX-DARI Component performs a direct access read.

5.2.3.3 The EX-DAWI Component performs a direct access write.

5.2.4 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

5.2.5 Get DCB Address (DAGDCB) Module locates and returns the address of a DCB in a linked list.

5.2.6 Close Data Set DCB (DACLOS) Module closes a DCB previously opened by DAOPN, delinks it from the DCB chain and deallocates the core it occupied.

5.3 The DB-DTAB Component deletes table sets of the Central Flow Control Data Base Set. It consists of the DBDTAB, DBERR, and DBWTMT Modules and Components.

5.3.1 The DBDTAB Module is the driver module for the DB-DTAB Component.

5.3.2 The DB-WTMT Component inserts and/or deletes entries in the table mapping table set.

5.3.3 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

6.0 The DB-BKUP Component controls and executes all off-line data base create and maintenance functions. It consists of the DB-BKUP, DB-STUP, DB-RCOY, and DB-CLUP Components.

6.1 The DB-BKUP Component creates backup tapes of the Central Flow Control Data Base Set. It consists of the DB-BKUP, DB-WAIM, DBLLMG, DB-DKTP, EX-AMBI, EX-SEI, EX-TBLI, EXTWTF, EX-TCLI, EX-RIDE, and EX-APEI components.

6.1.1 The DBBKUP Module checks the backup option, locks all updatable sets, calls DBDKIP to copy the data base from disk to tape, unlocks data sets and prepares the system for TLM processing or system shutdown.

6.1.2 The DB-WAIM Component is the interface between the Executive Subsystem and the Data Base Management Subsystem for obtaining the data base work area, critical address, and TLM dependent parameters.

6.1.2.1 The DBWAIM Module is the driver module for the DB-WAIM Component.

6.1.2.2 The EX-ETII Component returns data base work area address and size, data base update security mask and the message ID for a transaction.

6.1.2.3 The EX-IFAI Component locates the incore address of core resident data sets.

6.1.2.4 The EX-GSPI Component returns the address of a set.

6.1.2.5 The EX-LDAI Component enqueues a data area.

6.1.2.6 The EX-DARI Component performs a direct access read.

6.1.2.7 The EX-UDAI Component dequeues a data area.

6.1.2.8 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

6.1.2.9 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

6.1.2.10 The EX-SEI Component sends an error message to the operator.

6.1.2.11 The EX-APEI Component aborts the current program element.

6.1.2.12 The EX-DAWI Component performs a direct access write.

6.1.3 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

6.1.4 The DB-DKTP Component performs the copying of the data base from disc to tape.

6.1.4.1 The DBDKTP Module issues tape volume requests, then copies the data base block by block from disc to tape until the entire data base is copied. It also handles tape volume switching whenever necessary.

6.1.4.2 The EX-SEI Component sends an error message to the operator.

6.1.4.3 The EX-TWTI Component writes a block to tape.

6.1.4.4 The EX-TCLI Component performs control functions on tapes.

6.1.4.5 The EX-TDLI Component delays processing.

6.1.4.6 The EX-DARI Component performs a direct access read.

6.1.5 The EX-AMBI Component accepts a message block from the queue.

6.1.6 The EX-SEI Component sends an error message to the operator.

6.1.7 The EX-TDLI Component delays processing.

6.1.8 The EX-TWTI Component writes a block to tape.

6.1.9 The EX-TCLI Component performs control functions on tapes.

6.1.10 The EX-RTDI Component returns control to the dispatcher.

6.1.11 The EX-APEI Component aborts the current program element.

6.2 The DB-STUP Component performs the startup/startover functions of the on-line data base. It consists of the DB-STUP, DB-WAIM, DB-LCRS, DB-STOV, DB-LBKD, DB-LALK, DB-INSP, DB-DKTP, EX-AMBI, EX-SEI, EX-RTDI, EX-DAWI, EX-TWTI, EX-TCLI, EX-TRDI, EX-TDLI, and EX-RSSI Components.

6.2.1 The DBSTUP module accepts instructions, one at a time from the main console in the form of DBOP messages, extracts the instructions from the message checks for validity, performs the function as requested and prompts for next instruction, until the data base is ready for TLM processing.

6.2.2 The DB-WAIM Component is the interface between the Executive Subsystem and the Data Base Management Subsystem for obtaining the data base work area, critical address, and TLM depending parameters.

6.2.3 The DBLCRS Component loads the core resident sets in accordance with the table mapping table set.

6.2.3.1 The DBLCRS Module is the driver module for the DB-LCRS Component.

6.2.3.2 The EX-DARI Component performs a direct access read.

6.2.3.3 The EX-DAWI Component performs a direct access write.

6.2.3.4 The EX-IFAI Component locates the incore address of core resident data sets.

6.2.4 The DB-STOV Component performs multiple restrictions of before look saved by all TLM's that had not been completed at system failure.

6.2.4.1 The DBSTOV Module reads the before looks saved by each TLM and restores the before looks back to the data base, repeats the procedure for all before/after look sets that are still active, finally purge all before/after look sets.

6.2.4.2 The EX-SEI Component sends an error message to the operator.

6.2.4.3 The EX-DARI Component performs a direct access read.

6.2.4.4 The EX-DAWI Component performs a direct access write.

6.2.5 The DB-LBKD Component performs the function of restoring the data base onto disc from a back-up tape.

6.2.5.1 The DBLBKD Module reads from a back-up tape a block at a time, writes the block onto disc and unlocks the tape when it reaches end-of-file or end-of-tape.

6.2.5.2 The EX-TRDI Component reads a block from a tape.

6.2.5.3 The EX-SEI Component sends an error message to the operator.

6.2.5.4 The EX-DAWI Component performs a direct access write.

6.2.5.5 The EX-TCLI Component performs control functions on tapes.

6.2.6 The DB-LALK Component performs the functions of recapturing all after looks saved by TLM's since the last back-up of the data base, thus bringing the data base up to date.

6.2.6.1 The DBLALK module will read one after look record from a before/after looks tape, then update the data base according to information in the after look record. The process is repeated until end-of-file or end-of-tape, then the after looks tape will be unloaded.

6.2.6.2 The EX-TRDI Component reads a block from a tape.

6.2.6.3 The EX-TCLI Component performs control functions on tapes.

6.2.6.4 The EX-SEI Component sends an error message to the operator.

6.2.6.5 The EX-DAWI Component performs a direct access write.

6.2.6.6 The EX-DARI Component performs a direct access read.

6.2.7 The DB-INSP Component inspects the Central Flow Control flight record set for pointer and record numbering errors.

6.2.7.1 The DBINSP Module is the driver module for the DB-INSP Component.

6.2.7.2 The DB-GETR Component retrieves a flight record from the Central Flow Control flight record set on the initial retrieval call.

6.2.7.3 The DB-GIND Component retrieves the pointer to a flight record in the Central Flow Control flight record set.

6.2.7.4 The EX-DARI Component performs a direct access read.

6.2.8 The DB-DKTP Component performs the copying of the data base from disc to tape.

6.2.9 The EX-AMBI Component accepts a message block from the queue.

6.2.10 The EX-SEI Component sends an error message to the operator.

6.2.11 The EX-RTDI Component returns control to the dispatcher.

6.2.12 The EX-DAWI Component performs a direct access write.

6.2.13 The EX-TWTI Component writes a block to tape.

6.2.14 The EX-TCLI Component performs control functions on tapes.

6.2.15 The EX-TRDI Component reads a block from a tape.

6.2.16 The EX-TDLI Component delays processing.

6.2.17 The EX-RSSI Component returns control to startup or startover.

6.3 The DB-RCOV Component provides for recovery of the Central Flow Control Data Base Set after an aborted PE. It consists of DB-RCOV, EXSEI, EX-DARI, EX-DAWI, and DB-ERR Components.

6.3.1 The DBRCOV Module reads the before looks saved by a single TLM and applies the before looks back to the data base, then purges the before/after look set created by the TLM.

6.3.2 The EX-SEI Component sends an error message to the operator.

6.3.3 The EX-DARI Component performs a direct access read.

6.3.4 The EX-DAWI Component performs a direct access write.

6.3.5 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

6.4 The DB-CLUP Component controls and executes all data base clean up functions and the data base recovery procedure if required. It consists of the DB-CLUP, DB-LLMG, DB-RCOY, DB-ERR, EX-APEI, EX-LDAI, EX-DARI, EX-UDAI, EX-DAWC, EX-GDRI, EX-TWTI, EXDBSI, EX-TCLI, and EX-SEI Components and Modules.

6.4.1 The DBCLUP Module stores all leftover transactions that could not be done by the transaction load module due to unforseen difficulties.

6.4.2 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

6.4.3 The EX-DAWI Component performs a direct access write.

6.4.4 The DB-RCOV Component nullifies the data base updates performed by a single TLM by restoring the data base to the status before the TLM was dispatched.

6.4.5 The EX-DARI Component performs a direct access read.

6.4.6 The DBERR Module processes all error messages encountered by the Data Base Management Subsystem.

6.4.7 The EX-TWTI Component writes a block to tape.

6.4.8 The EX-APEI Component aborts the current program element.

6.4.9 The EX-DBSI Component logs a data base statistic to disk.

6.4.10 The EX-LDAI Component enqueues a data area.

6.4.11 The EX-TCLI Component performs control functions on tapes.

6.4.12 The EX-DARI Component performs a direct access read.

6.4.13 The EX-SEI Component sends an error message to the operator.

6.4.14 The EX-UDAI Component dequeues a data area.

7.0 The DB-APIN Component performs all interface functions between the data base and other subsystems who utilize the data base flight record files. It consists of the DB-GTFR and DB-UPFR Components.

7.1 The DB-GTFR Component is the interface between the data base and the Application Subsystem for the retrieval of flight record data items. It consists of the DBGTFR, DB-GETR, DB-GETE, DBFILT, and DBEXTR, and DB-LLMG Modules and Components.

7.1.1 The DBGTFR module is the driver module for the DB-GTFR Component.

7.1.2 The DB-LLMG Component handles all lock, lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

7.1.3 The DB-GETR Component retrieves a flight record from the Central Flow Control flight record set on an initial call for flight records.

7.1.4 The DB-GETE Component retrieves subsequent flight records from the Central Flow Control Flight Record Set.

7.1.5 The DB-FILT Component filters flight records passed back to the caller based on the requestors input parameters.

7.1.5.1 The DBFILT Module is the driver module for the DB-FILT Component.

7.1.5.2 The DBTFLT Module filters flight records passed back to the caller depending on the time of day given by the caller.

7.1.6 The DBEXTR Module extracts data items from a flight record as specified by the caller.

7.2 The DB-UPFR Component is the interface between the data base and the Application Subsystem for changes to flight records of the Central Flow Control Data Base Set. It consists of the DB-UPFR, DB-GETR, DB-LLMG, DB-INST, and DB-CHGR Modules and Components.

7.2.1 The DBUPFR Module is the driver module for the DB-UPFR component.

7.2.2 The DB-LLMG Component handles all lock , lock reduction, unlock and lock test requests and maintains the integrity of the lock request queue.

7.2.3 The DB-GETR Component retrieves a flight record from the Central Flow Control Flight Record Set on an initial call for flight records.

7.2.4 The DB-CHGR Component changes flight records data items in the Central Flow Control Flight Record Set.

7.2.5 The DB-INST Component inserts a flight record into the Central Flow Control Flight Record Set.

2.2.1.4 Application (AP) Subsystem

a. Purpose. The purpose of the Application (AP) Subsystem is to process the 26 messages specified in the CPFS document entitled "Central Flow Control Computer Program Specifications: Volume II, Application Program Specification." These messages allow inquiry and updating of information contained in the CFC Data Base and the performing of predictions and simulations in order to assist Air Traffic Control Systems Command Center (ATCSCC) personnel in performing their duties of ensuring an equitable and uniform flow of air traffic within the continental United States.

b. Subsystem Overview. The Application Subsystem consists of a set of 26 Transaction Load Modules (TLMs) that process the 26 messages specified for the initial implementation of the CFC Software System. Figure AP-1.0 is an overview diagram for the Application Subsystem showing external interfaces. The Application Subsystem has direct interfaces with only two other subsystems: the Executive Subsystem, and the Data Base Management Subsystem. As shown in Figure 2.2.1-12, all interfaces with these two subsystems will be through common external interface components used by all 26 TLMs. The purpose of using these interface components is to minimize the impact on the AP caused by changes to either the Executive or the Data Base Management Subsystems.

The Application Subsystem accepts a message from the Executive Subsystem via the AP-GETMSG Component. The message is parsed and internal control information is extracted by the AP-INPUT Component. The

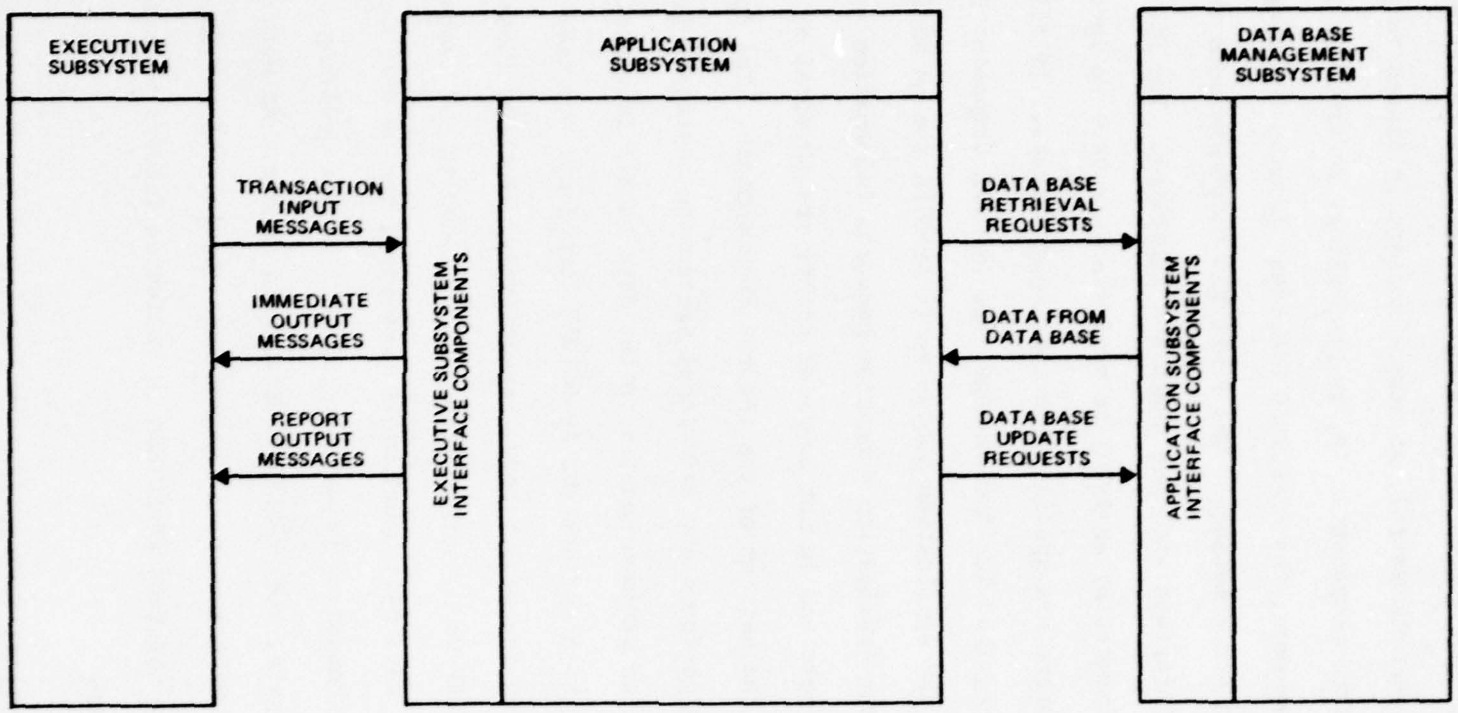


Figure 2.2.1-12. AP-1.0 Application Subsystem (AP) Overview

AP-PUTMSG Component echoes the original message back to the sender. If processing was successful, an accept message is issued to the sender via the AP-PUTMSG Component. If, in processing, an error is detected in the input message, the AP-PUTMSG Component is used to issue a reject message and an error message. The AP-PUTMSG Component uses the AP-SENDIM Component to interface with the Executive Subsystem. The DB-TABT Component is invoked by AP-INPUT to verify and convert to internal form the control parameters specified in the input message. If flight record retrieval is required for the message, the DB-GTFR Component is used. Requests from the Application Subsystem to DB-GTFR are on an informational level, i.e., the Application Subsystem requests information from the flight record sets and is not aware of either the physical or logical structures of the sets or of the records they contain. The Application Subsystem then performs any additional processing necessary to set up output reports or perform updates to the data base. If reports are to be generated, the AP-OUTHDR and AP-OUTDAT Components are used to output header lines and the report body, respectively. Both of these components use the AP-SENDRP Component to issue report lines to the Executive Subsystem. If data base updating is necessary, the DB-UPFR Component or the DB-SETT Component is called. DB-UPFR causes updating of the flight record sets, and DB-SETT causes table update. As with DB-GTFR, requests to DB-UPFR are on an informational basis. If the data base is successfully updated, AP-PUTMSG is called to issue a transaction-complete message.

c. Functional Description. The Application Subsystem is divided into four functional classes of TLMs according to the types of CFC messages. They are:

- List messages that perform retrieval and formatting of information in the data base

- Count messages that perform retrieval, counting processes, and formatting of information contained in the data base

- Simulation messages that perform simulations using information contained in the data base

- Update messages that perform alterations of and additions to data base records

All four classes of TLMs perform data base retrieval and interface with the Executive Subsystem for resource management and I/O services. With the exception of the FADF, FADP, QFLW, and QFLZ messages (belonging to the simulation class) only the update messages perform data base updating. The four functional classes are divided into a set of ten major components (Figure 2.2.1-12). These components perform the processing necessary to handle the 26 messages. Table 2.2.1-1 shows the relationship between the four functional classes, the ten major components, and the transaction messages handled by each major component. The major components use the following AP components to perform common functions:

- AP-INPUT to retrieve the input message and parse it into fields and elements

- AP-PUTMSG to assemble and output status and error messages

- AP-ERROR to issue error messages

Table 2.2.1-1. Relationship of Message Classes, Major Components, and Transaction Messages Handled

FUNCTIONAL CLASS	MAJOR COMPONENT	TRANSACTION MESSAGES HANDLED
1 LIST	1 LISTRA	LISA, LISD
	2 LISTFL	LIFP
	3 LISTAB	CAPL, GAEL
2 COUNT	4 REPDMD	DEMA, DEMD, DESA, DESD, DLDY
	5 COMFIX	FIXL
3 SIM	6 PERSIM	ARRD, FADF, FADP, FADT, QFLW, QFLZ
4 UPDATE	7 ALTFLT	ACTV, CXSD, INHB, RS
	8 UPDTAB	CAPS, GAES
	9 ADDFLP	FP, FPSD
	10 ADDEPT	DM

- AP-OUTHDR to format and output report header lines
- AP-OUTDAT to format and output report data lines

The following components are used by the Application Subsystem to interface with the Data Base Management Subsystem:

- DB-WAIM to initialize a Data Base work area
- DB-GTFR to retrieve flight record information
- DB-UPFR to update flight record information
- DB-TABT to retrieve data base table information
- DB-SETT to update data base table information
- DB-CLUP to provide data base cleanup at end of message processing

The following components are used by the Application Subsystem to interface with the Executive Subsystem:

- EX-AMBI to accept an input message
- EX-GDRI to acknowledge abnormal message processing termination
- EX-RTDI to acknowledge normal message processing termination
- EX-SMBI to send a message block

d. <u>Major Components.</u> The ten major components of the Application Subsystem are:

- AP-LISTRA to list arrival/departure traffic at a specified pacing airport or center
- AP-LISTFL to list the flight plan of specified aircraft for all legs requested
- AP-LISTAB to list table (general aviation or capacity) for all specified pacing airports
- AP-REPDMD to report current/future arrival/departure statistics for specified airport or center
- AP-COMFIX to compute arrival fix loading for specified fix or pacing airport
- AP-PERSIM to perform arrival demand, quota flow, or fuel advisory simulation for the specified pacing airport. It generates specifie reports and performs data base updates if necessary.
- AP-ALTFLT to activate, inhibit, or cancel specified flights
- AP-UPDTAB to update table (general aviation or capacity) for specified pacing airport
- AP-ADDFLP to add flight plan for specified aircraft
- AP-ADDEPT to add actual departure time for specified aircraft

These components are described in Sections 2.2.1.4.1 through 2.2.1.4.10, respectively.

e. Input. The Application Subsystem receives its input from two sources. The messages originating from the ATCSCC and the ARTCCs are passed to the Application Subsystem by the Transaction Control Component of the Executive Subsystem. All flight record data and information relating to airport capacities, fixes, etc., are retrieved through the services of the Data Base Management Subsystem. These inputs are considered to be included in the following data sets, respectively:

- Transaction Control Component Data Set - EXTCCS
- Central Flow Control Data Base Set - CFCDBS

f. Output. Outputs of the Application Subsystem are either in the form of reports to be sent back to the ATCSSC or updates to the CFC Data Base. These outputs are placed in the following data sets, respectively:

- Transaction Control Component Data Set - EXTCCS
- Central Flow Control Data Base Set - CFCDBS

2.2.1.4.1 List Traffic (AP-LISTRA) Component

a. Purpose. The List Traffic Component lists arrival or departure traffic at a pacing airport or all pacing airports in a center. This component processes both the LISA and the LISD messages.

b. Functional Description. The AP-LISTRA Component uses either a LISA or LISD message communicated from the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the LISA or LISD message. The header lines for the report to be generated are then output. For each pacing airport specified, all flights that satisfy the qualifier information are retrieved from the data base. If required, the center code for the departure airport is retrieved and the flight status (planned or active) is determined. The list of flights is then sorted in ascending arrival time or departure time (depending upon whether a LISA or LISD message was received), and the report body is output.

c. Input. A LISA or LISD message is retrieved from the Executive Subsystem in the following set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The output from AP-LISTRA is in the form of immediate messages sent back to the originator of the message via the Executive Subsystem, and a report on the departure or arrival flights.

The output is placed in the following Executive Subsystem interface set: Transaction Output Message Queue Set - EXTOMS.

2.2.1.4.2 List Flight Plan (AP-LISTFL) Component

a. Purpose. The List Flight Plan Component lists the flight plan of a specified aircraft for all legs requested. This component processes the LIFP message.

b. Functional Description. The AP-LISTFL Component uses a LIFP message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If the message information is incorrect, a diagnostic message is sent to the originator of the LIFP message. Flight plan information is retrieved from the data base either for the flight legs that arrive or depart the specified pacing airport or for all flight legs that arrive or depart any pacing airport. The header line for the report is output and then the report lines for the requested legs are output.

c. Input. A LIFP message is retrieved from the Executive Subsystem in the following set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The output from AP-LISTFL consists of immediate messages that are sent back to the originator of the LIFP message and a report on all flight legs requested. The output is placed in the following Executive Subsystem interface set: Transaction Output Message Queue Set - EXTOMS.

2.2.1.4.3 List Table (AP LISTAB) Component

a. Purpose. The List Table Component lists either the general aviation or capacity table for a specified pacing airport or all pacing airports. This component processes the CAPL and GAEL messages.

b. Functional Description. The AP-LISTAB Component uses either a CAPL or a GAEL message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the CAPL or GAEL message via the Executive Subsystem. For CAPL, today and normal values of the capacities for the specified pacing airport or all pacing airports are retrieved from the data base. For GAEL today and normal values of the general aviation estimates for the specified pacing airport or center are retrieved from the data base. The report table is built from the retrieved values for the time period specified. The header and data lines for the report are then output.

c. Input. A CAPL or GAEL message is retrieved from the Executive Subsystem in the following set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following source: Table Set - TABSET (CFCDBS).

d. Output. The output from the AP-LISTAB Component is in the form of immediate messages that are sent back to the originator of the CAPL or GAEL message and a report on the requested table. The output is placed in the following Executive Subsystem interface set: Transaction Output Immediate Message Set - EXTOMS.

2.2.1.4.4 Report Demand (AP-REPDMD) Component

a. Purpose. The Report Demand Component provides a report on current or future arrival or departure statistics for a specified airport or center. This component processes the DEMA, DEMD, DESA, DESD, and DLDY messages.

b. Functional Description. The AP-REPDMD Component uses a DEMA, DEMD, DESA, DESD, or DLDY message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the input message. The report header lines are then output. Flight record information for a pacing airport, non-pacing airport, or all pacing airports in a center is then retrieved and counter values for the report period are incremented. General aviation estimates for a pacing airport or center are retrieved and GA counters are incremented, if necessary. The count values obtained are then used to generate and output the demand report.

c. Input. A DEMA, DEMD, DESA, DESD, or DLDY message is retrieved from the Executive Subsystem in the following set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The output from AP-REPDMD is in the form of immediate messages that are sent back via the Executive Subsystem to the originator of the message and a report on the demand expected at the given airport or center. The output is placed in the following Executive Subsystem interface set: Transaction Output Message Queue Set - EXTOMS

2.2.1.4.5 Compute Fix Loading (AP-COMFIX) Component

a. Purpose. The Compute Fix Loading Component reports the loading for a given arrival fix or all arrival fixes for a specified pacing airport. This component processes the FIXL message.

b. Functional Description. The AP-COMFIX Component uses a FIXL message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the FIXL message. The header information for the report is then output. Flight record information for

the pacing airport specified or implied by the fix ID is retrieved from the data base. For each fix, hour counters are incremented for flights that originate from departure airports that use the particular arrival fix. The report data is then output.

c. Input. A FIXL message is retrieved from the Executive Subsystem in the following set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The output from AP-COMFIX is in the form of immediate messages that are sent back to the originator of the FIXL message and a report on the fix loading for the specified fix or pacing airport. The output is placed in the following Executive Subsystem Interface Set: Transaction Output Message Queue Set - EXTOMS.

2.2.1.4.6 Perform Simulation (AP-PERSIM) Component

a. Purpose. The Perform Simulation Component performs arrival demand, quota flow, or fuel advisory simulation for a specified pacing airport. This component is invoked in response to six CFC simulation messages (ARRD, FADF, FADP, FADT, QFLW, and QFLZ).

b. Functional Description. The Perform Simulation Component provides decoded and converted input message data to the Simulation

Subsystem which it invokes, and then provides report generation and data base update functions as required. The input message is retrieved from a message buffer, then parsed into elements and repeating elements which are verified and converted for internal control information. If the message information is incorrect, a diagnostic message is sent to the originator of the simulation message. Otherwise, the Simulation Subsystem (SI) is then invoked to perform one of the three types of simulation. The Simulation Subsystem is responsible for retrieving flight records and table information from the data base. After simulation completion, the reports peculiar to the simulation type are generated. If either the FADF or FADP message is being processed, the flight records which received controlled departure times by the Simulation Subsystem are updated in the data base. For the FADF, FADP, QFLW, and QFLZ messages, data in the Simulation Continue Table Set are updated.

c. Input. The Perform Simulation Component retrieves the simulation input message from the Executive Subsystem in the following data set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The AP-PERSIM Component outputs immediate messages that are sent back to the originator of the input message. Reports are generated and are message specific. The report types are:

- ARRD - Arrival Delay
- FADF - Landing Capacities, General Aviation Estimates, Quota Flow, Arrival Delay, and Energy Conservation
- FADP - Landing Capacities, General Aviation Estimates, Quota Flow, Arrival Delay, and Energy Conservation
- FADT - Quota Flow, Arrival Delay, and Energy Conservation Test
- QFLW - Quota Flow
- QFLZ - Quota Flow and Origin Center

The output messages and reports are placed in the following Executive Subsystem interface data set: Transaction Output Message Queue Set - EXTOMS.

For the FADF and FADP messages, the flight records assigned flow control departure delays are updated in the data base. Flight records are updated in one of the following data sets:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)

The Simulation Continue Table Set (CO) is updated for the FADF, FADP, QFLW, and QFLZ messages.

2.2.1.4.7 Alter Flight Status (AP-ALTFLT) Component

a. Purpose. The Alter Flight Status Component modifies the departure status of one or more specified flights. This component processes the ACTV, INHB, CXSD, and RS input messages which activate, inhibit, and cancel flights in the data base.

b. Functional Description. The AP-ALTFLT Component uses an alter flight status message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the message. For the specified aircraft identification, all flights that satisfy the qualifier information are retrieved from the data base. The activity or departure date mask is then modified in each flight record to reflect the new flight status. The updated flight records are stored back in the data base. A message is then issued to the originator stating that the transaction processing has been completed.

c. Input. The Alter Flight Status Component uses the ACTV, INHB, CXSD, or RS input message communicated from the Executive Subsystem in the following data set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The AP-ALTFLT Component sends back to the originator immediate type messages via the Executive Subsystem. The output messages are placed in the following data set: Transaction Output Message Queue Set - EXTOMS.

The altered flight records are output to the data base in one of the following sets:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)

2.2.1.4.8 Update Table (AP-UPDTAB) Component

a. Purpose. The Update Table Component updates either the General Aviation Table or the Landing Capacity Table of the data base. Only "today" values are updated, not the "normal" values. This component processes both the CAPS and GAES messages.

b. Functional Description. The AP-UPDTAB Component uses either a CAPS or GAES message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information

is incorrect, a diagnostic message is sent to the originator of the message. The report header lines are then output. The "today" and "normal" table values are retrieved from the data base tables. The "today" values are modified by the input message table values. A report of the table values is then generated and the new "today" values are updated in the data base.

c. Input. The Update Table Component uses the CAPS or GAES input message communicated via the Executive Subsystem in the following data set: Transaction Input Message Queue Set - EXTIMS.

The data base table values are obtained from the following source: Table Set - TABSET (CFCDBS).

d. Output. The AP-UPDTAB Component sends back to the originator via the Executive Subsystem immediate type messages and a report of the new or latest general aviation or landing capacity values. The output is placed in the following Executive Subsystem interface data set: Transaction Output Message Queue Set - EXTOMS.

2.2.1.4.9 Add Flight Plan (AP-ADDFLP) Component

a. Purpose. The Add Flight Plan Component creates a new flight plan and adds it to the data base. This component processes both the FPSD and FP messages.

b. Functional Description. The AP-ADDFLP Component uses either an FPSD or FP message communicated via the Executive Subsystem. The

message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the message. A check is made of the data base to see if the flight to be created already exists. If it does, a message is sent to the originator. Otherwise, input message flight record information is added to the flight record being built. The departure, arrival, and activity date masks are created and placed in the flight record. The flight record is then added to the data base. A message is sent to the originator stating that the transaction processing is completed.

c. Input. The ADD Flight Plan Component retrieves the FPSD or FD input message from the Executive Subsystem in the following data set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. Output. The AP-ADDFLP outputs immediate type messages which are placed in the following data set: Transaction Output Message Queue Set - EXTOMS.

The new flight record is added to one of the following sets:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)

2.2.1.4.10 Add Departure Time (AP-ADDEPT) Component

a. Purpose. The Add Departure Time Component adds the actual departure time, received from an input message, to a flight record associated with a specified flight plan leg. This component processes the DM message request.

b. Functional Description. The AP-ADDEPT Component uses a DM message communicated via the Executive Subsystem. The message fields are verified and converted to internal control information using data base conversion tables. If message information is incorrect, a diagnostic message is sent to the originator of the DM message. All flight records satisfying the qualifier information are retrieved from the data base. The flight record having a proposed gate time of departure within a set time limit of the actual departure time is the flight record to be updated. The actual departure time is stored in the flight record and the flight record is returned to the data base.

c. Input. The Add Departure Time Component uses the DM input message communicated via the Executive Subsystem in the following data set: Transaction Input Message Queue Set - EXTIMS.

Data base information is obtained from the following sources:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)
- Table Set - TABSET (CFCDBS)

d. <u>Output.</u> The AP-ADDEPT Component sends back immediate type messages to the originator via the Executive Subsystem. The output messages are placed in the following data set: Transaction Output Message Queue Set - EXTOMS.

The updated flight record is output to the data base in one of the following sets:

- OAG Flight Record Set - OFRS (CFCDBS)
- Nonscheduled Flight Record Set - NFRS (CFCDBS)

The Visual Table of Contents for the AP Subsystem is shown in Figure 2.2.1-13.

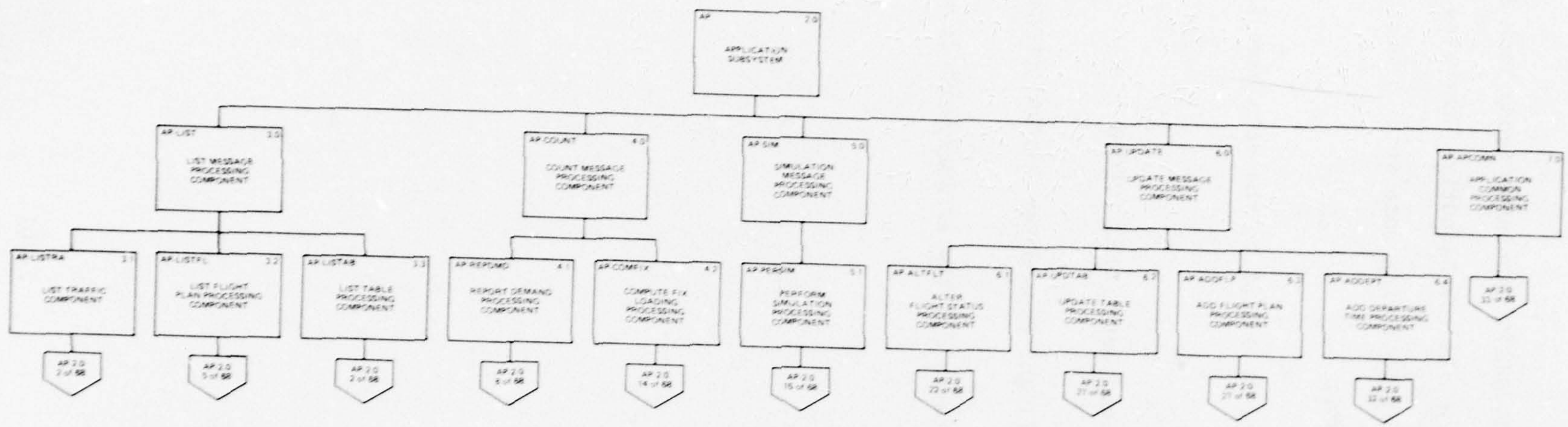


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (1 of 68)

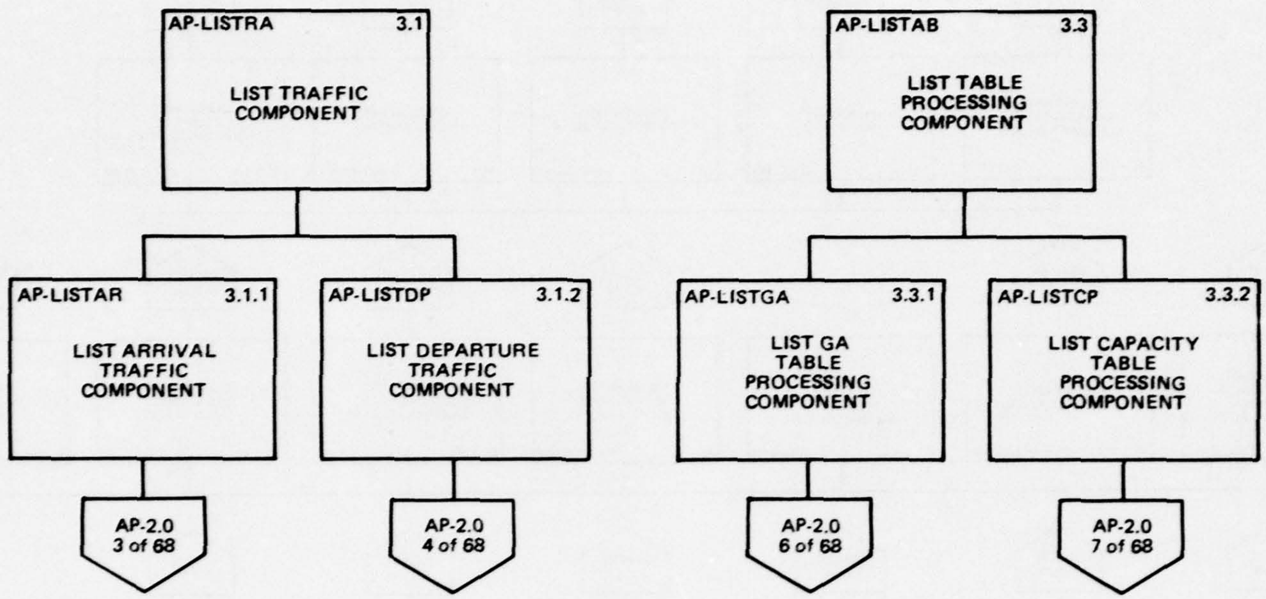


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (2 of 68)

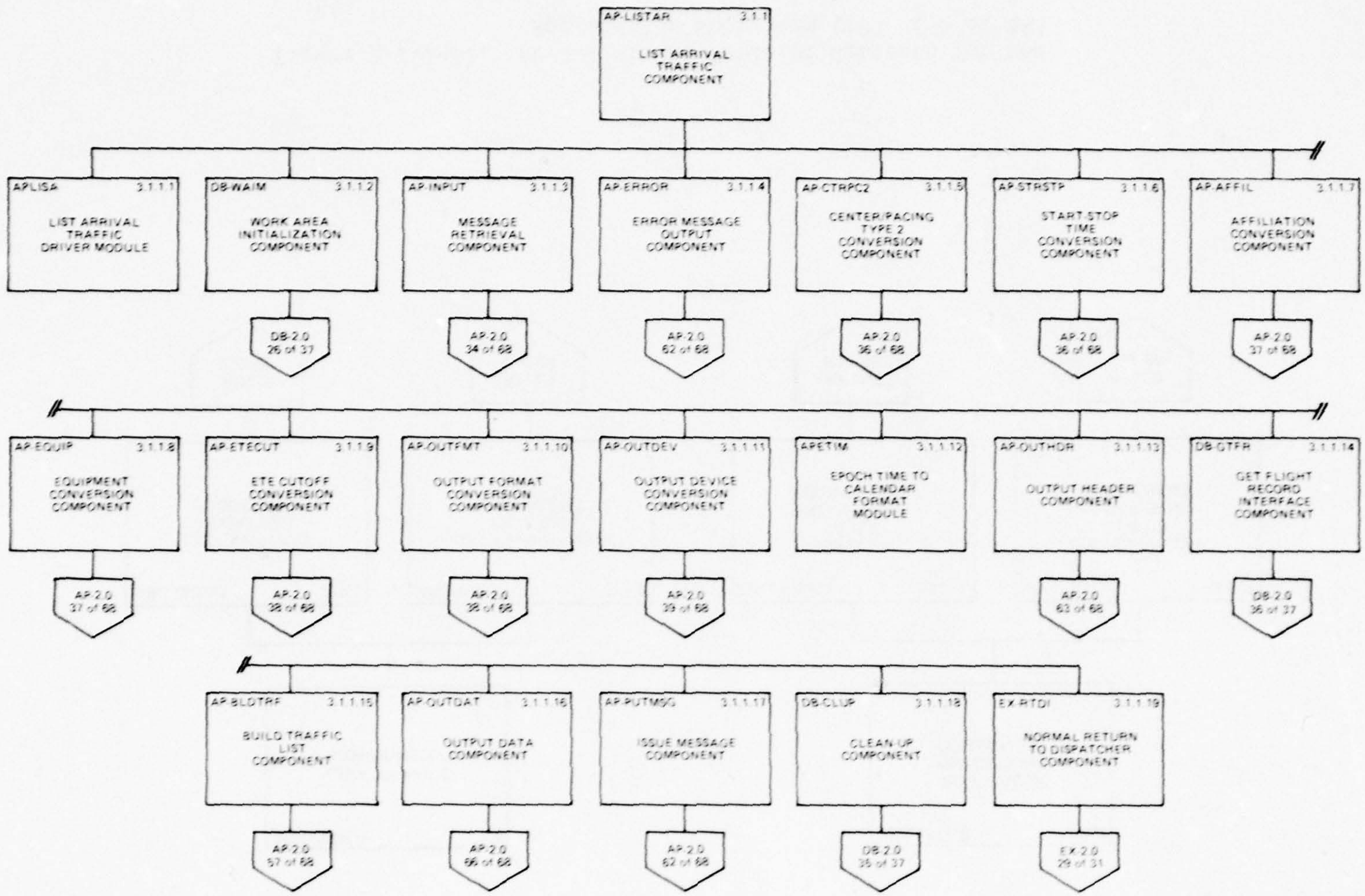


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (3 of 68)

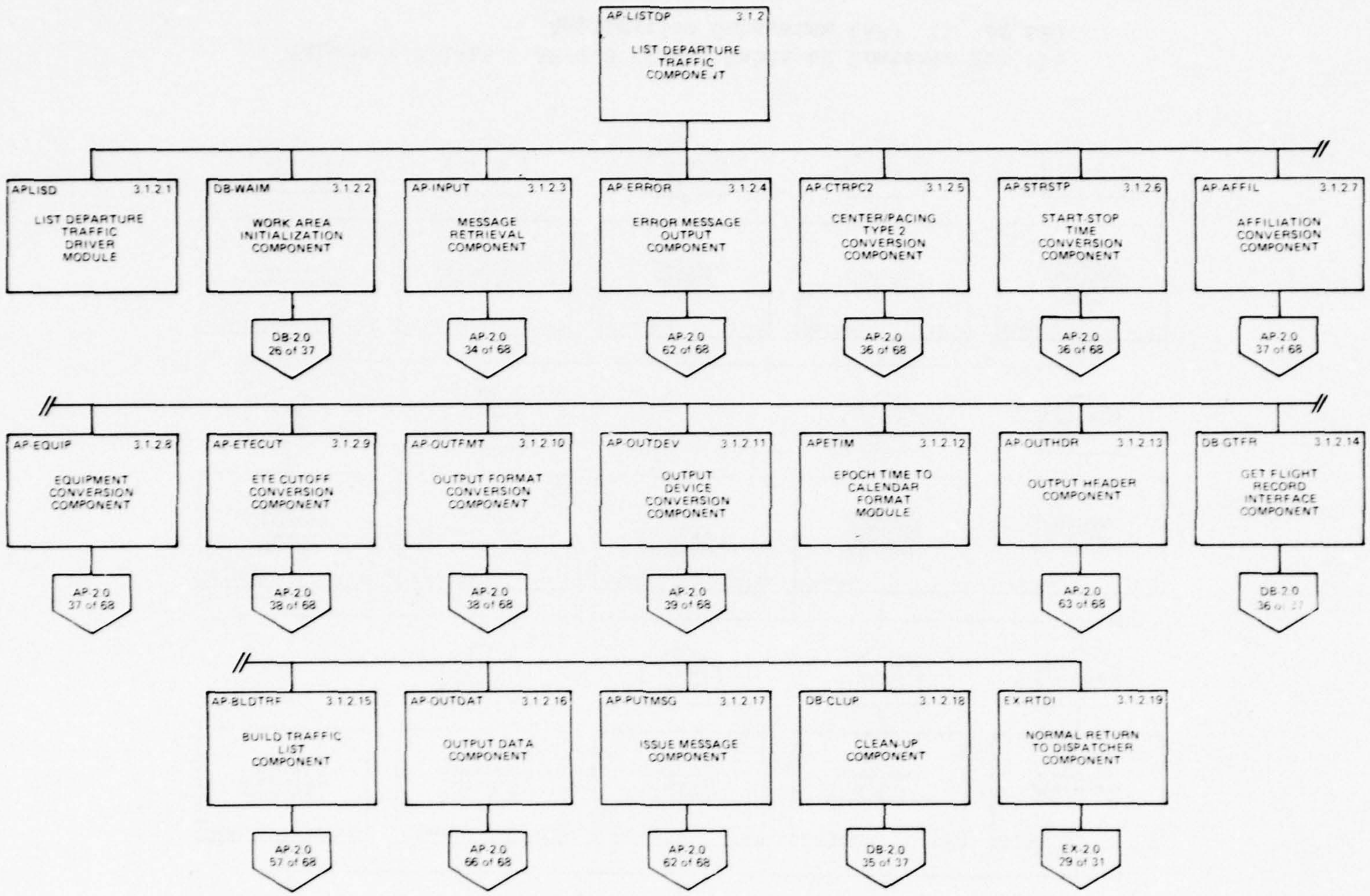


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (4 of 68)

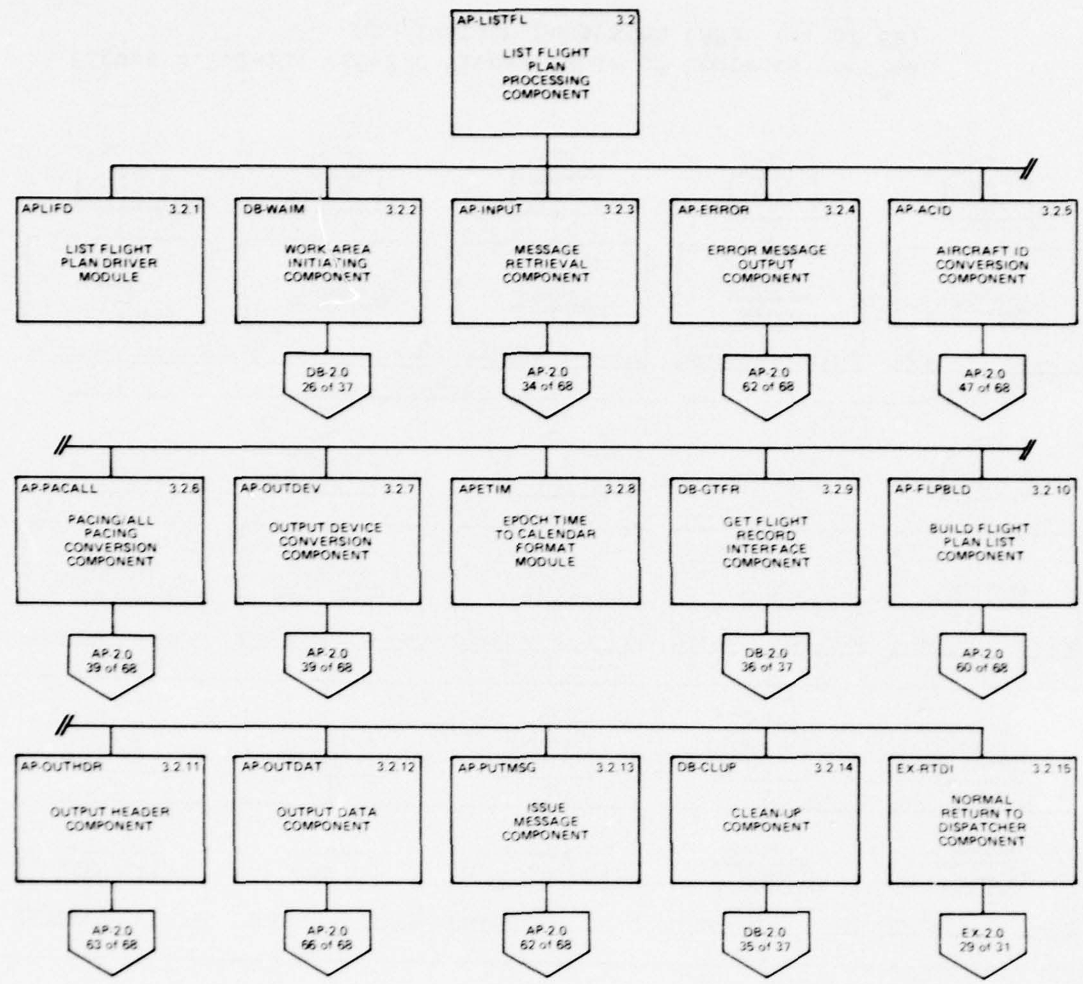


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (5 of 68)

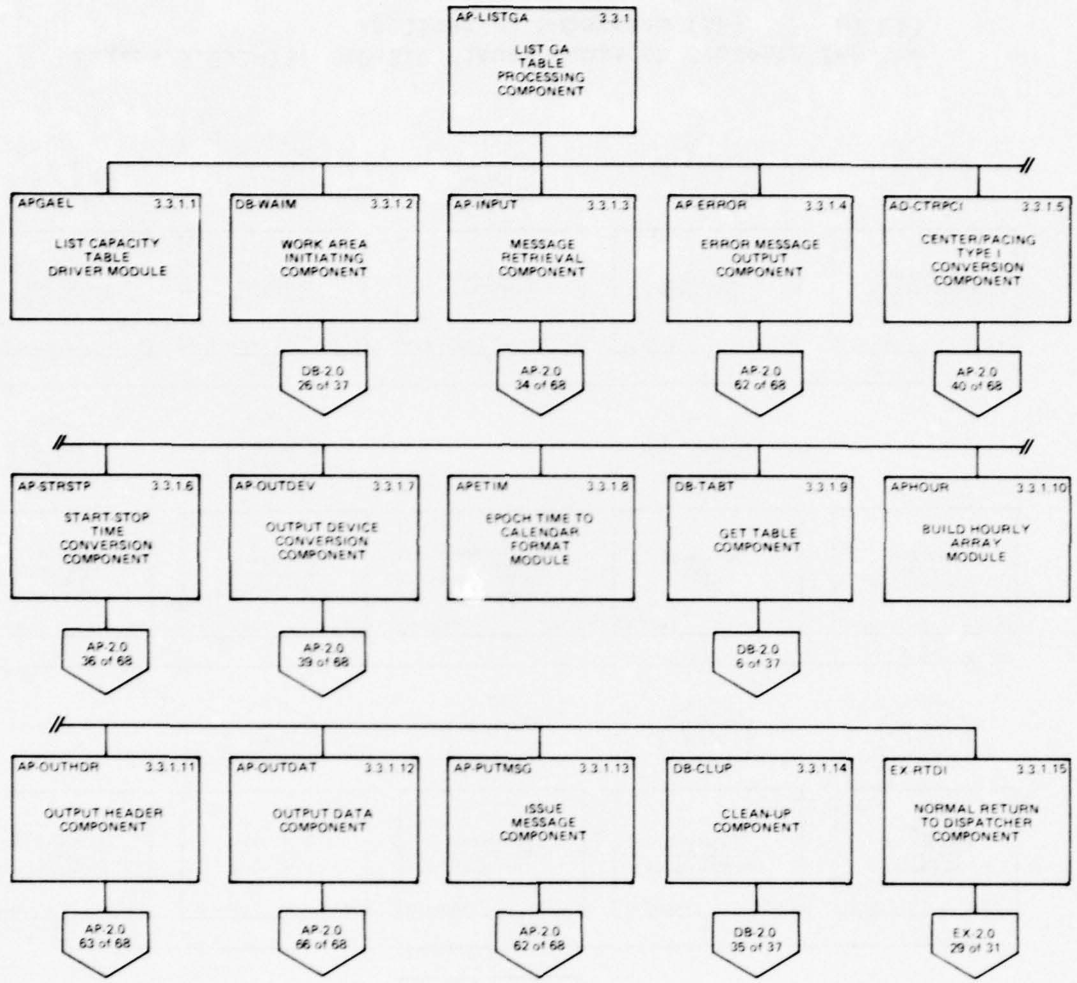


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (6 of 68)

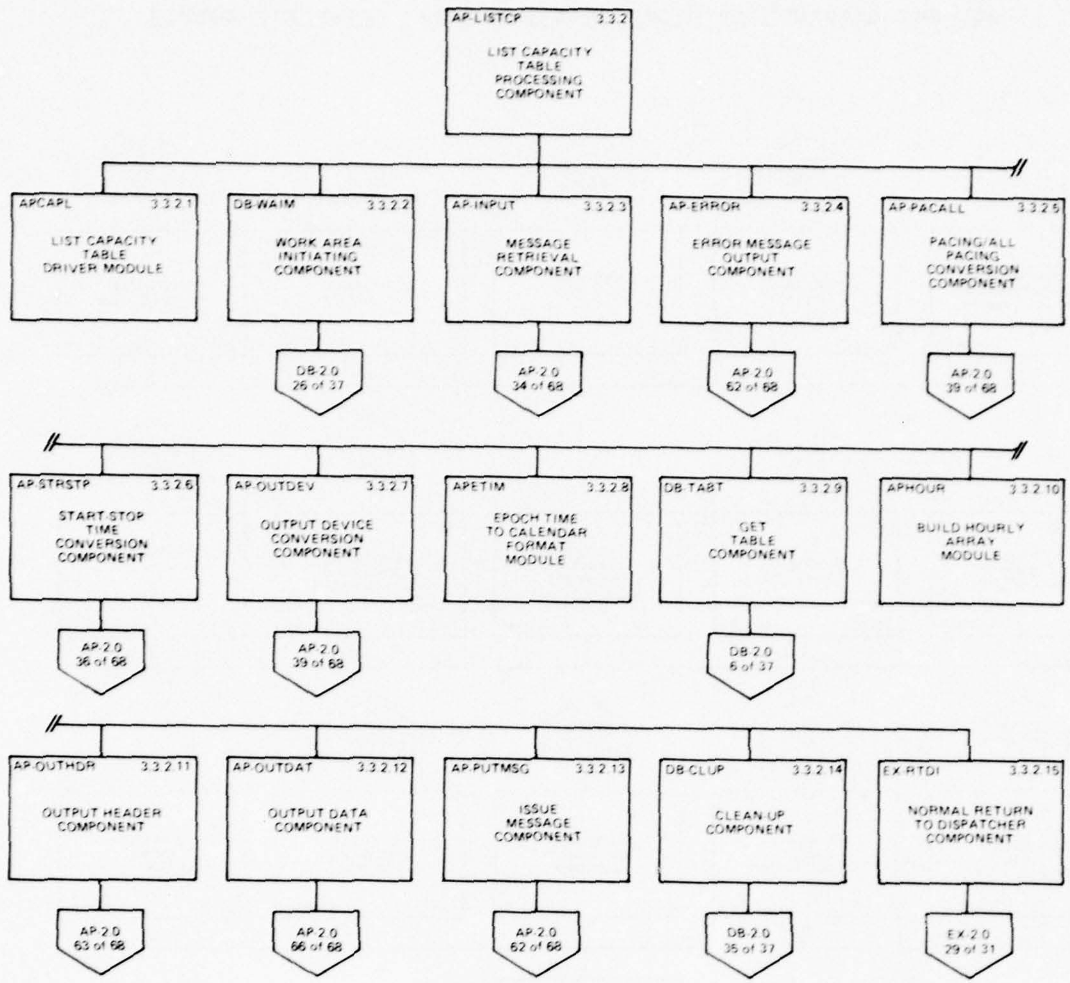


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (7 of 68)

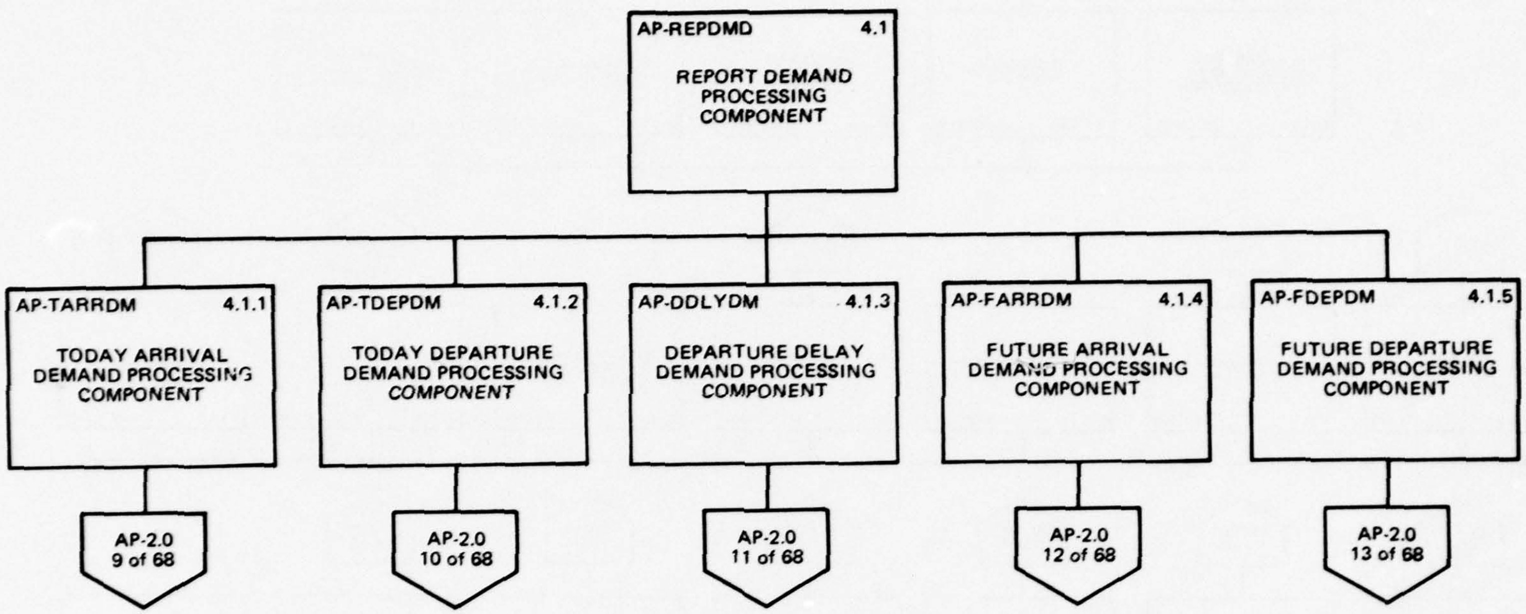


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (8 of 68)

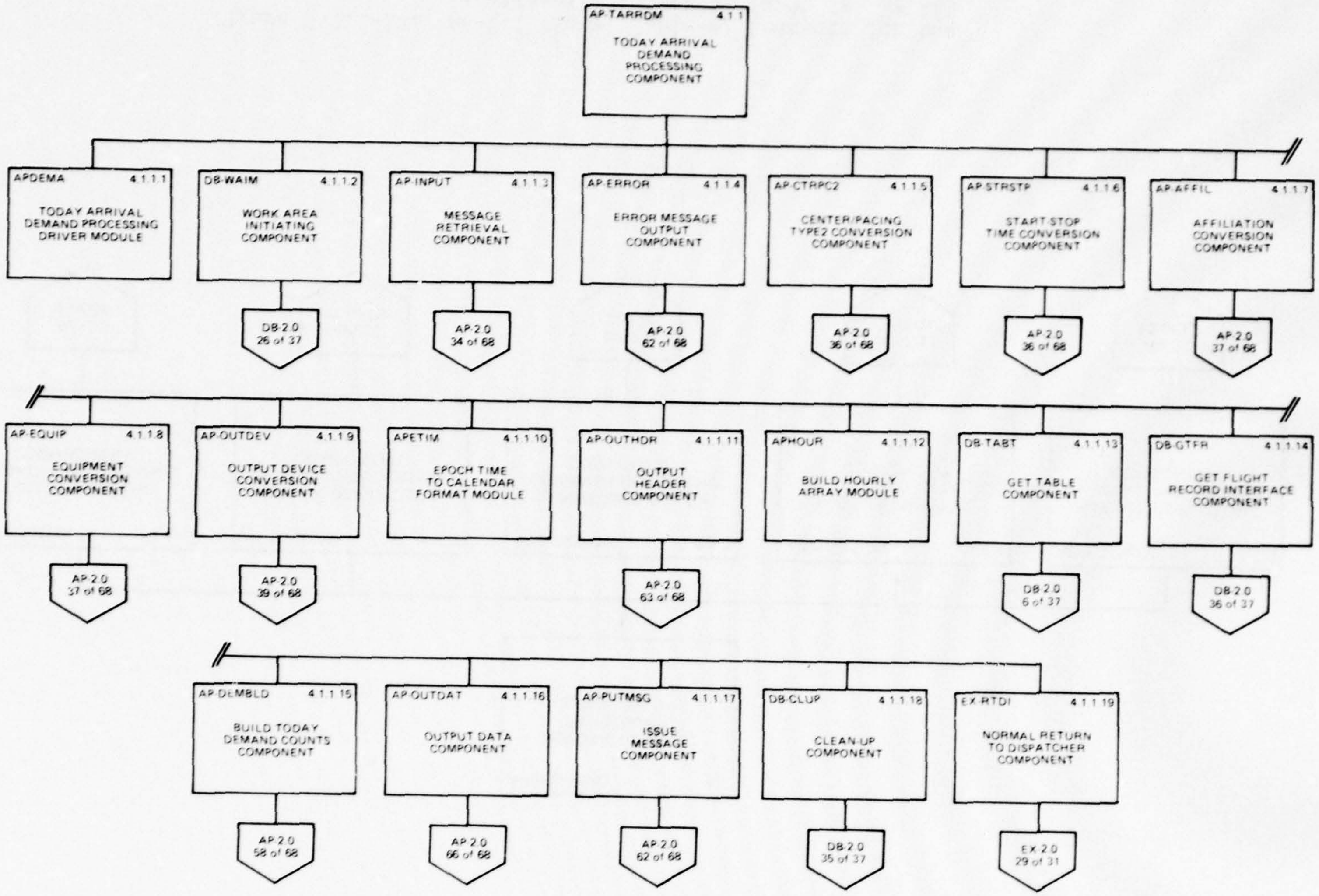


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (9 of 68)

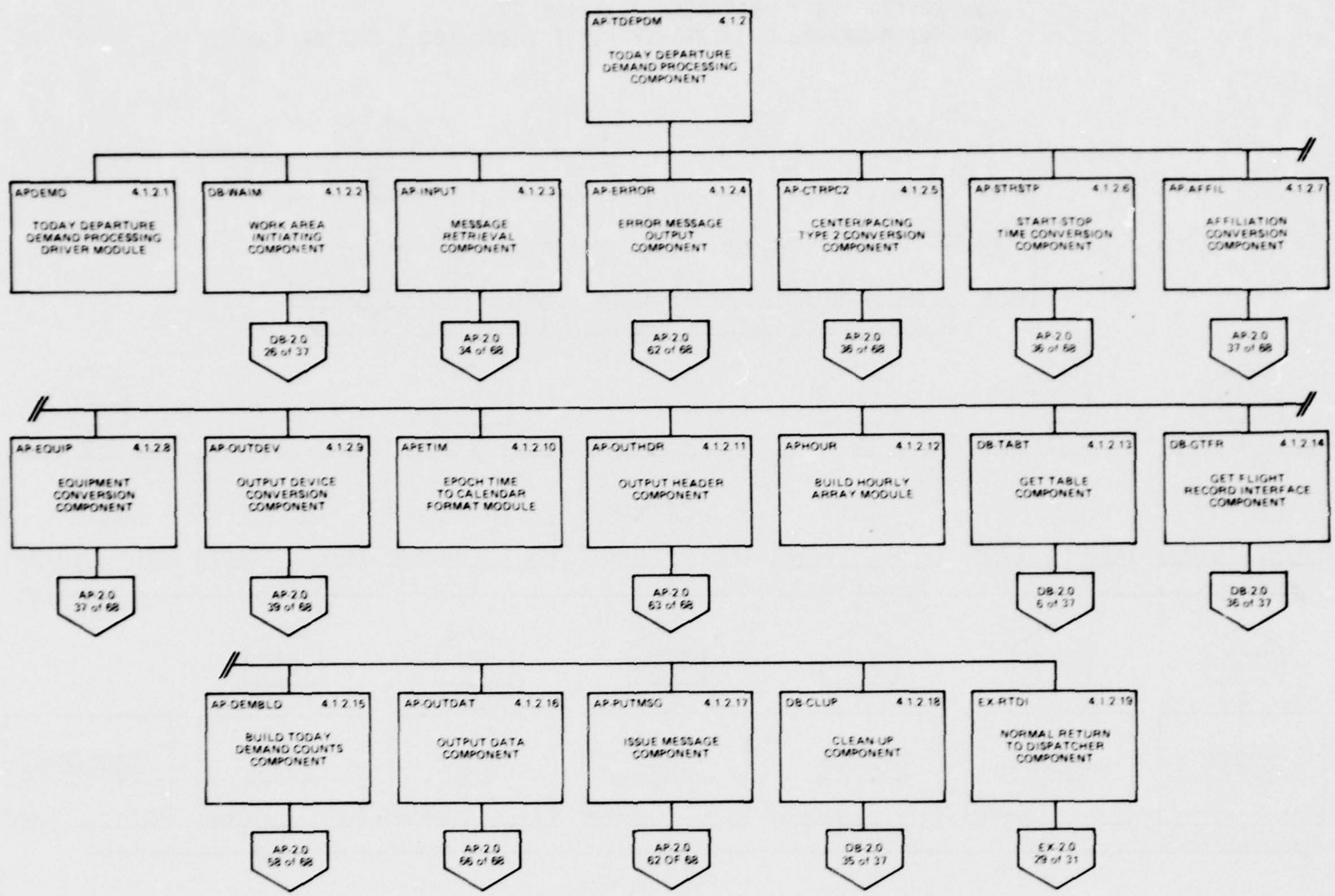


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (10 of 68)

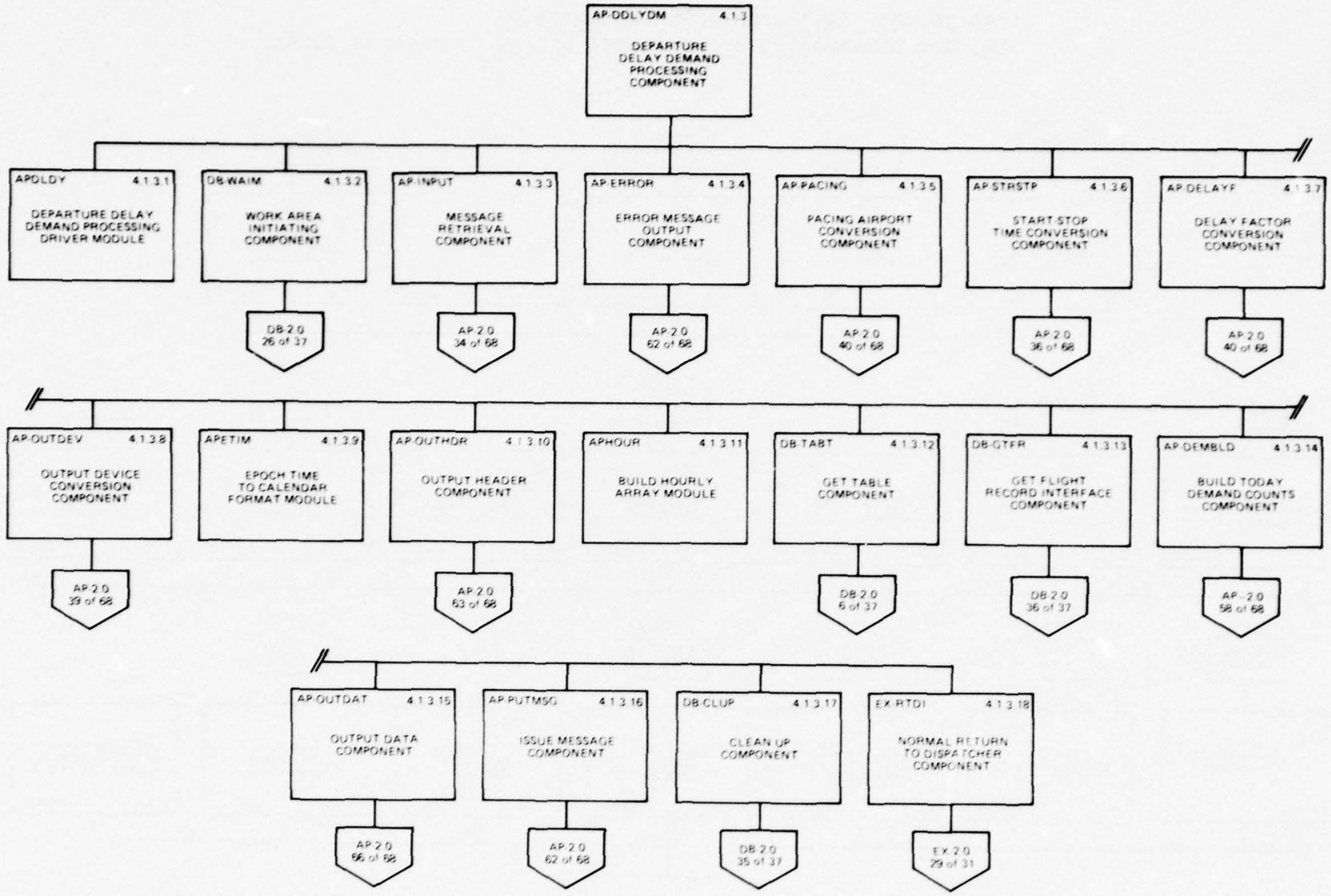


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (11 of 68)

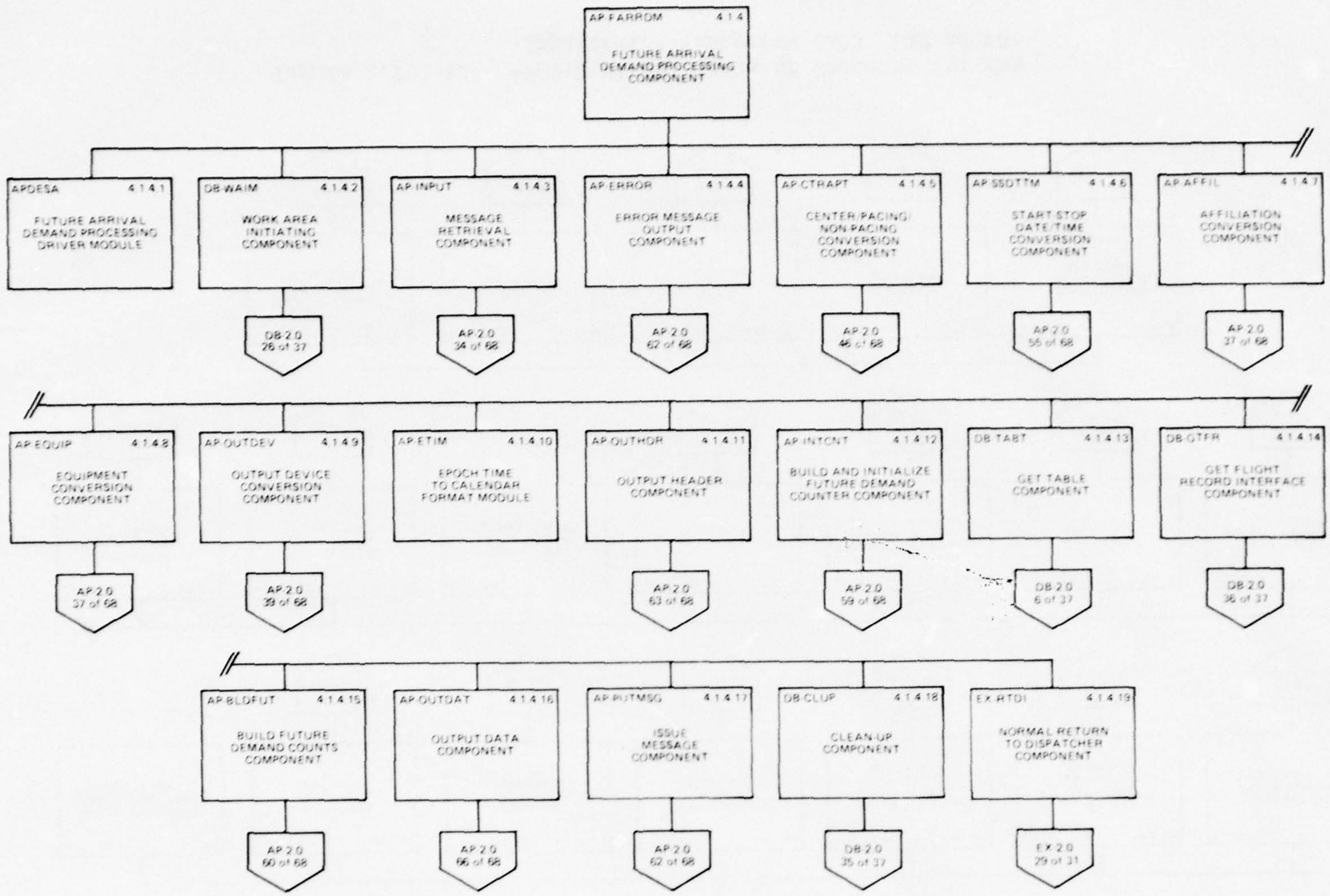


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (12 of 68)

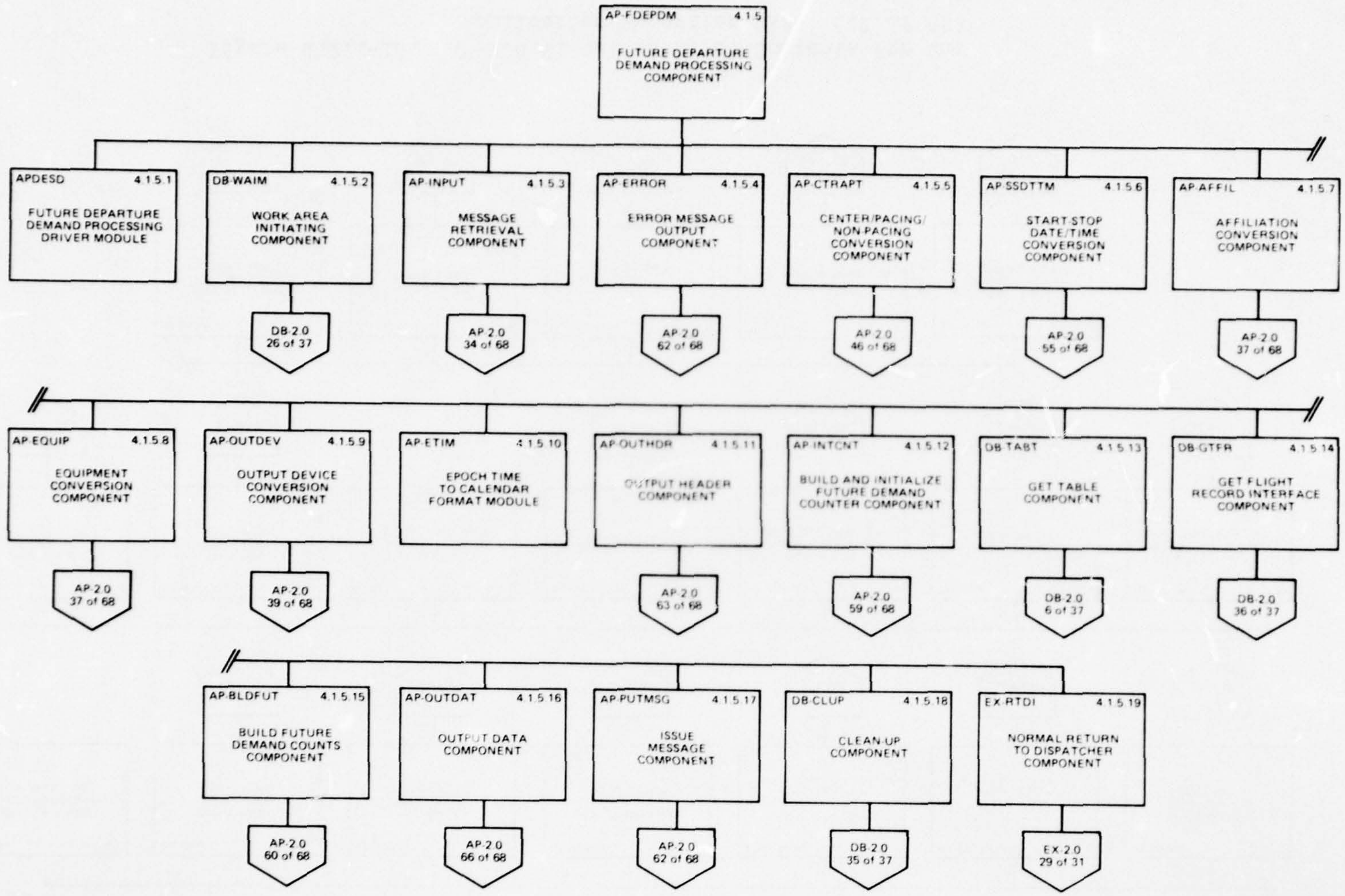


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (13 of 68)

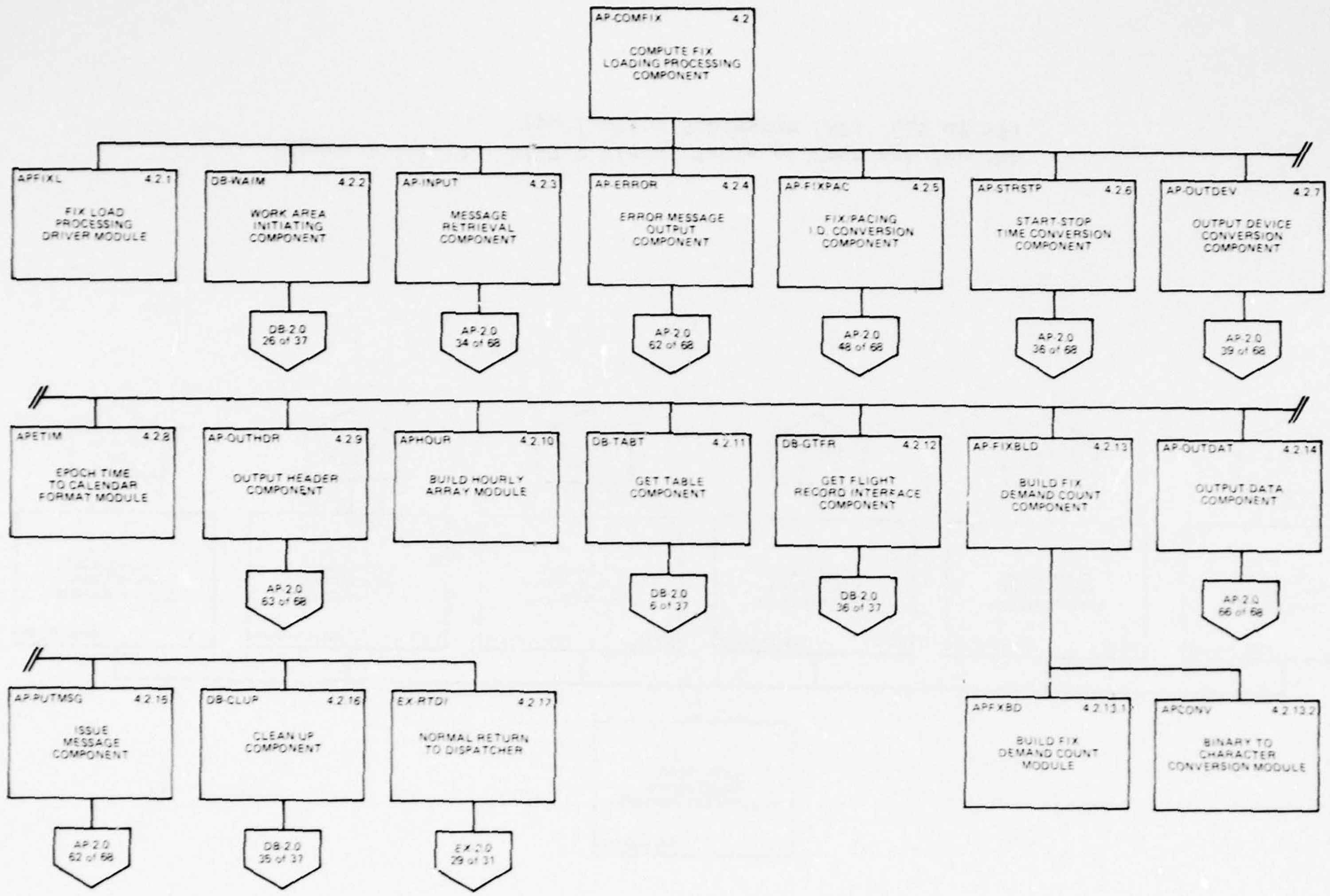


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (14 of 68)

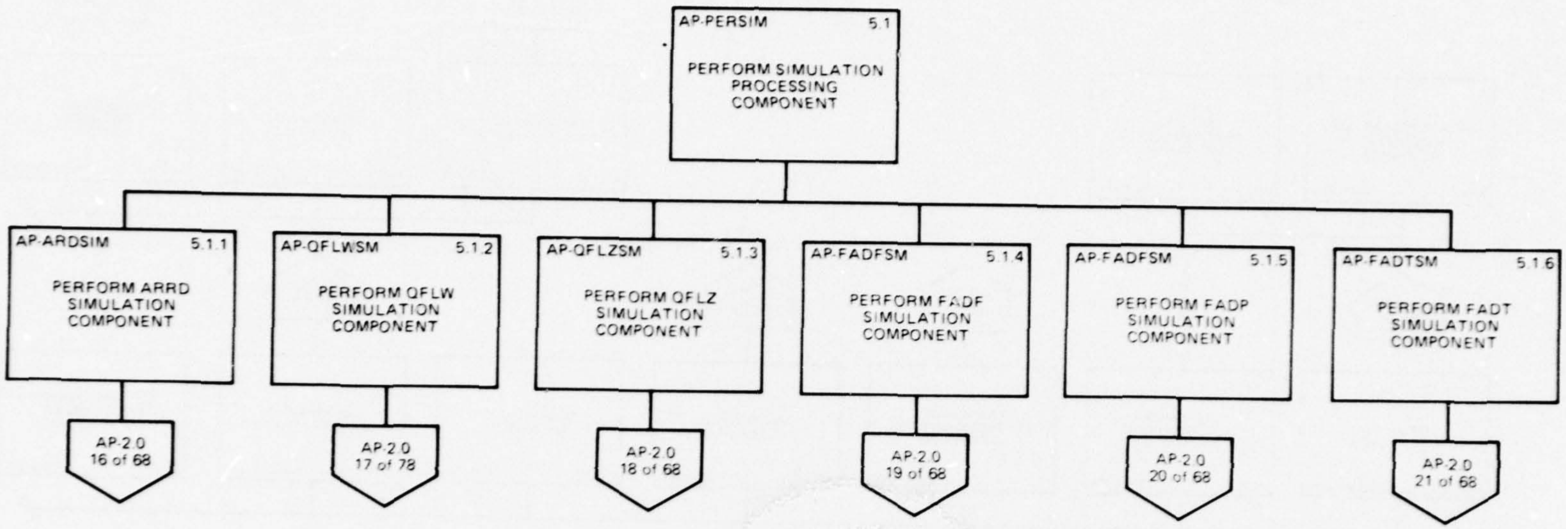


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (15 of 68)

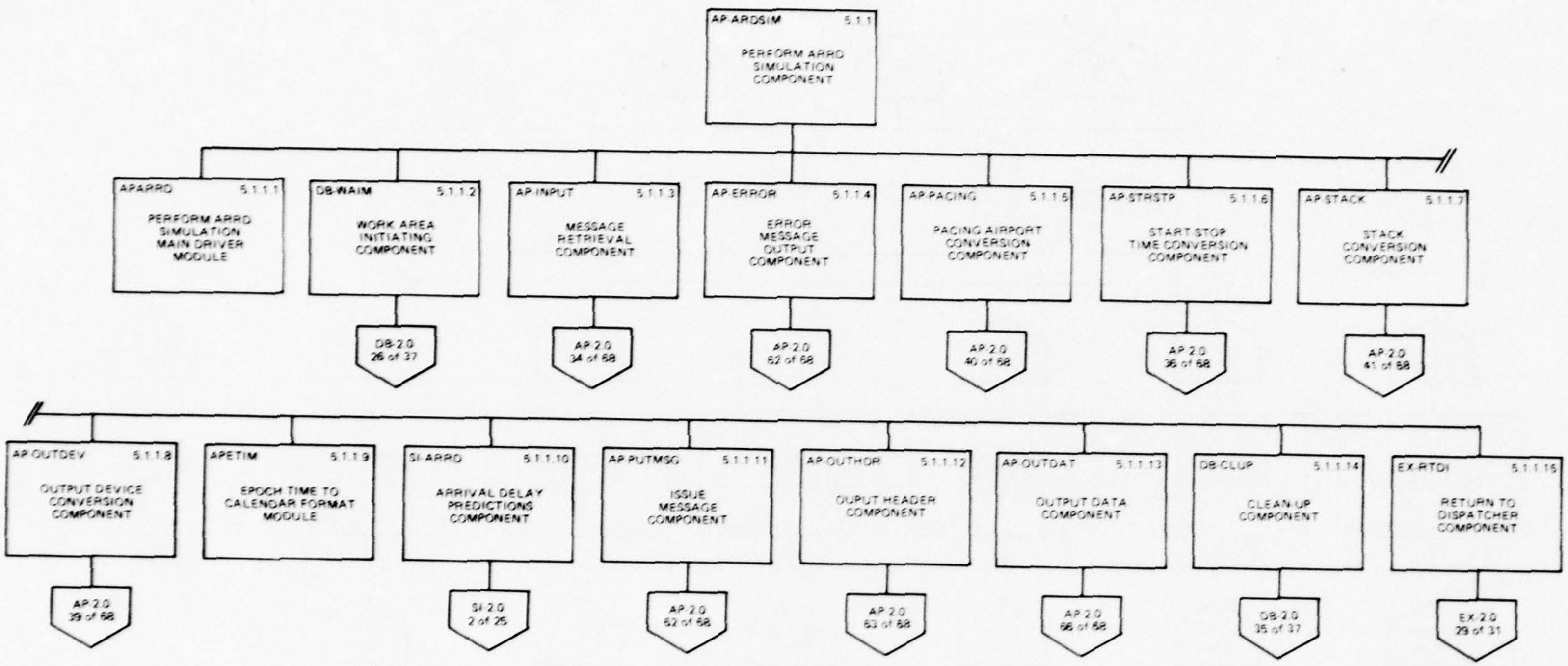


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (16 of 68)

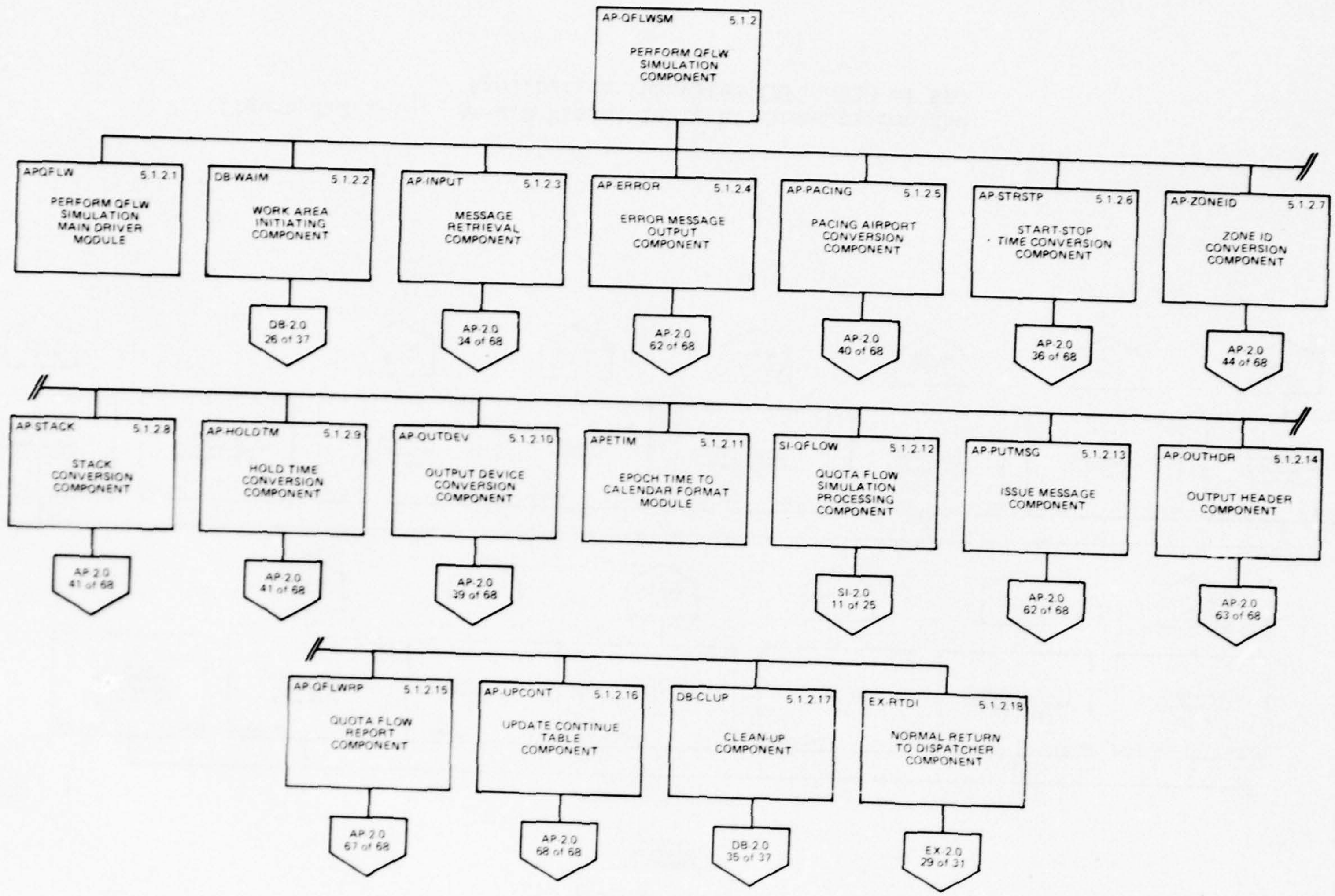


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (17 of 68)

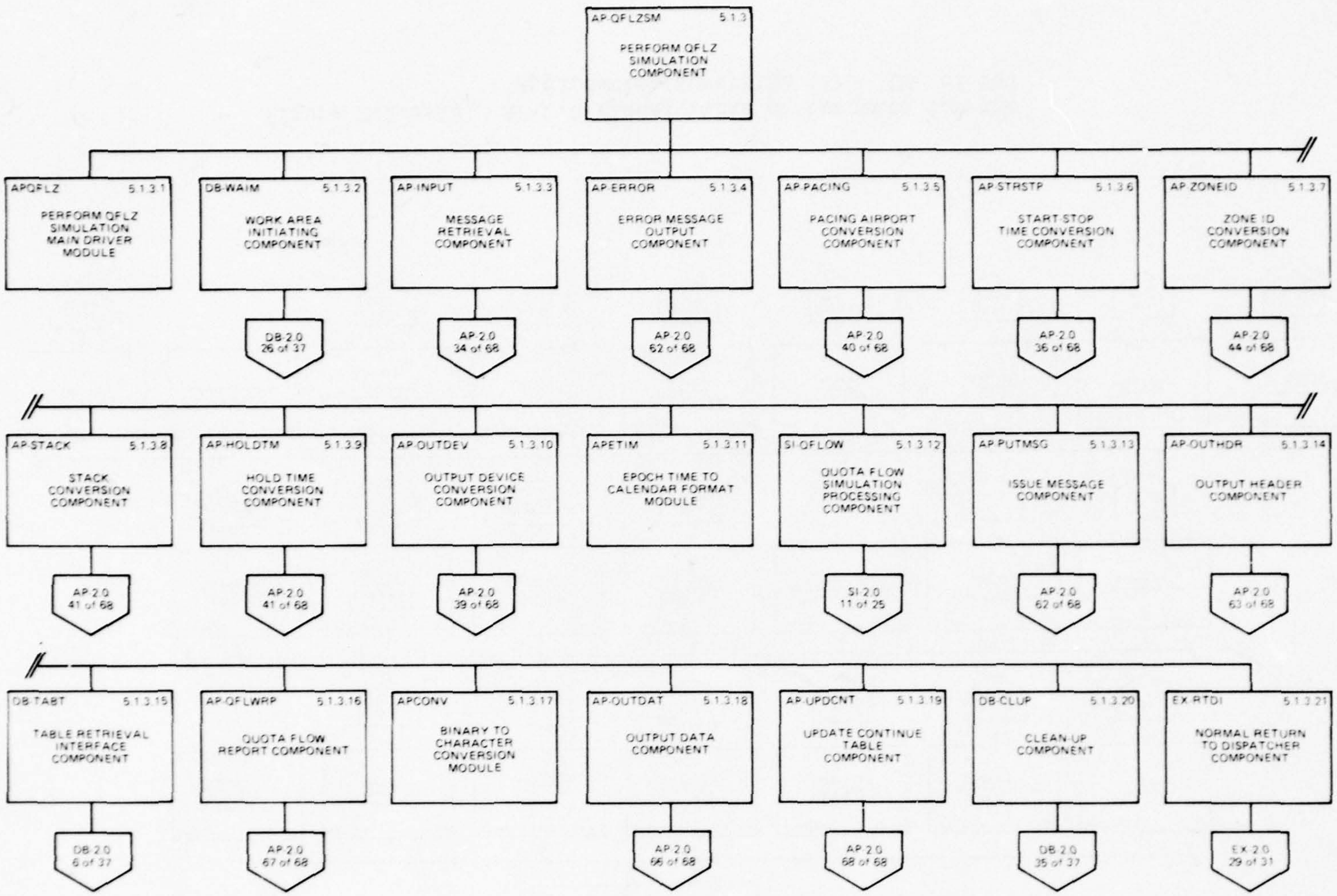


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (18 of 68)

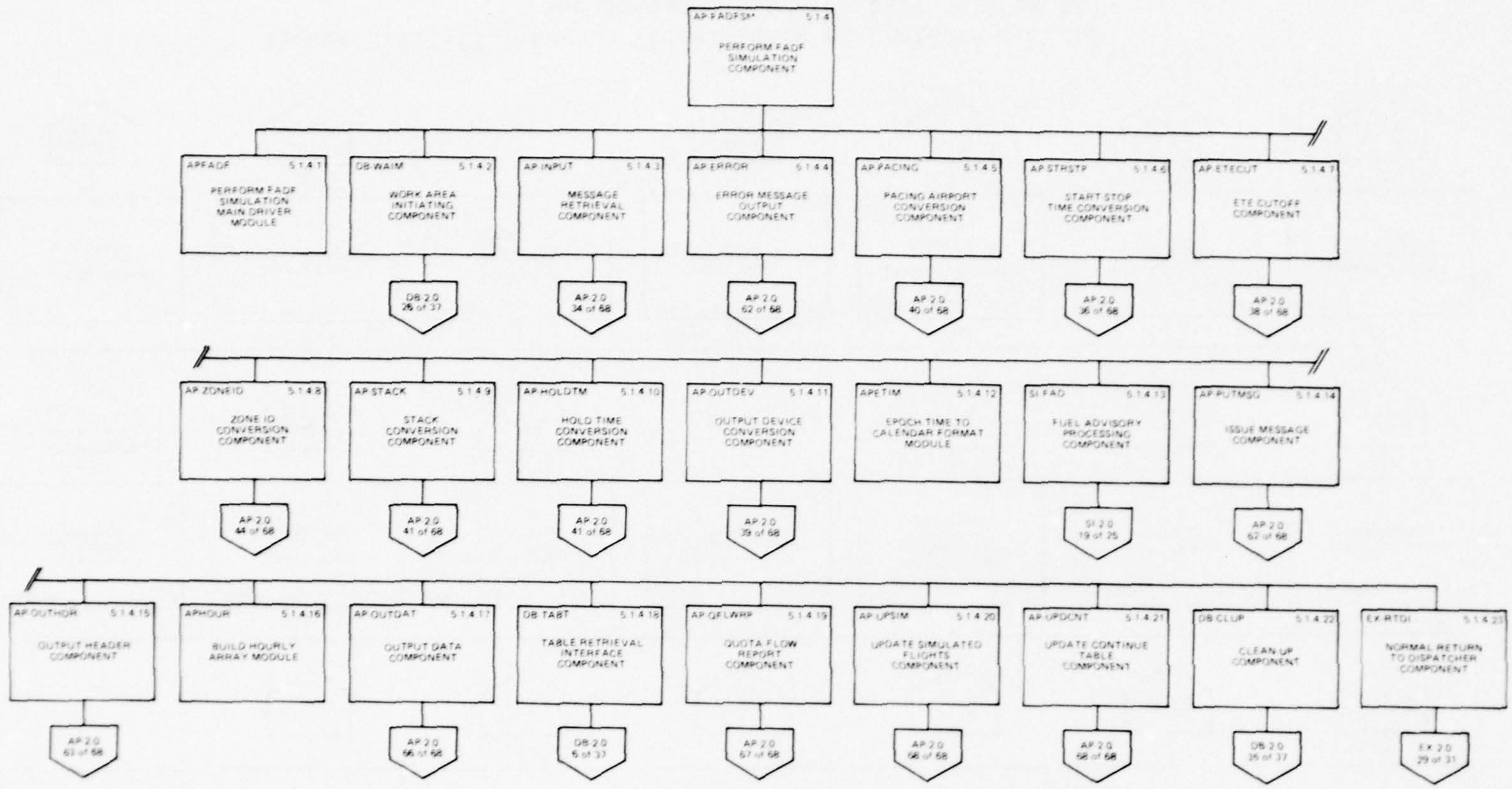


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (19 of 68)

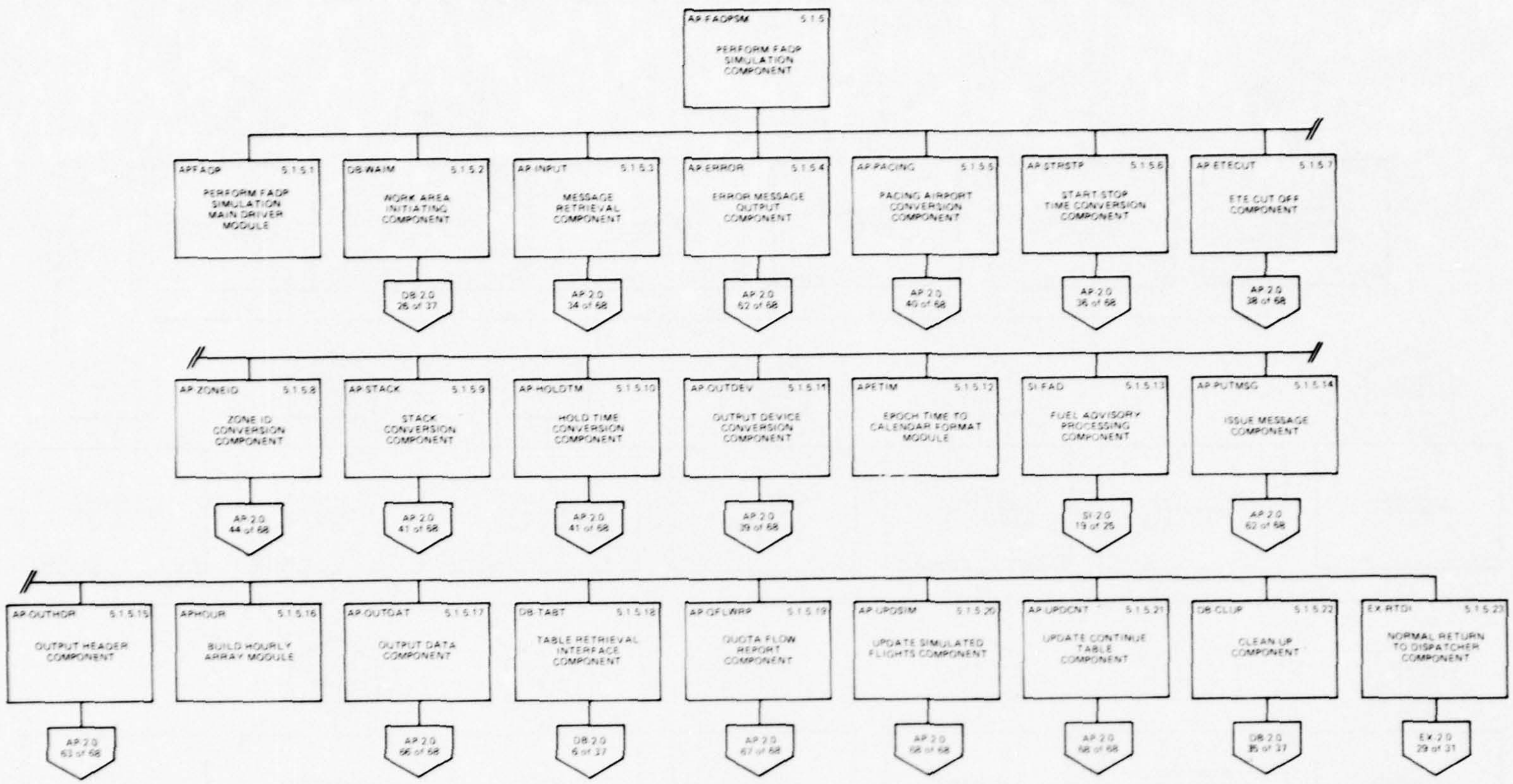


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (20 of 68)

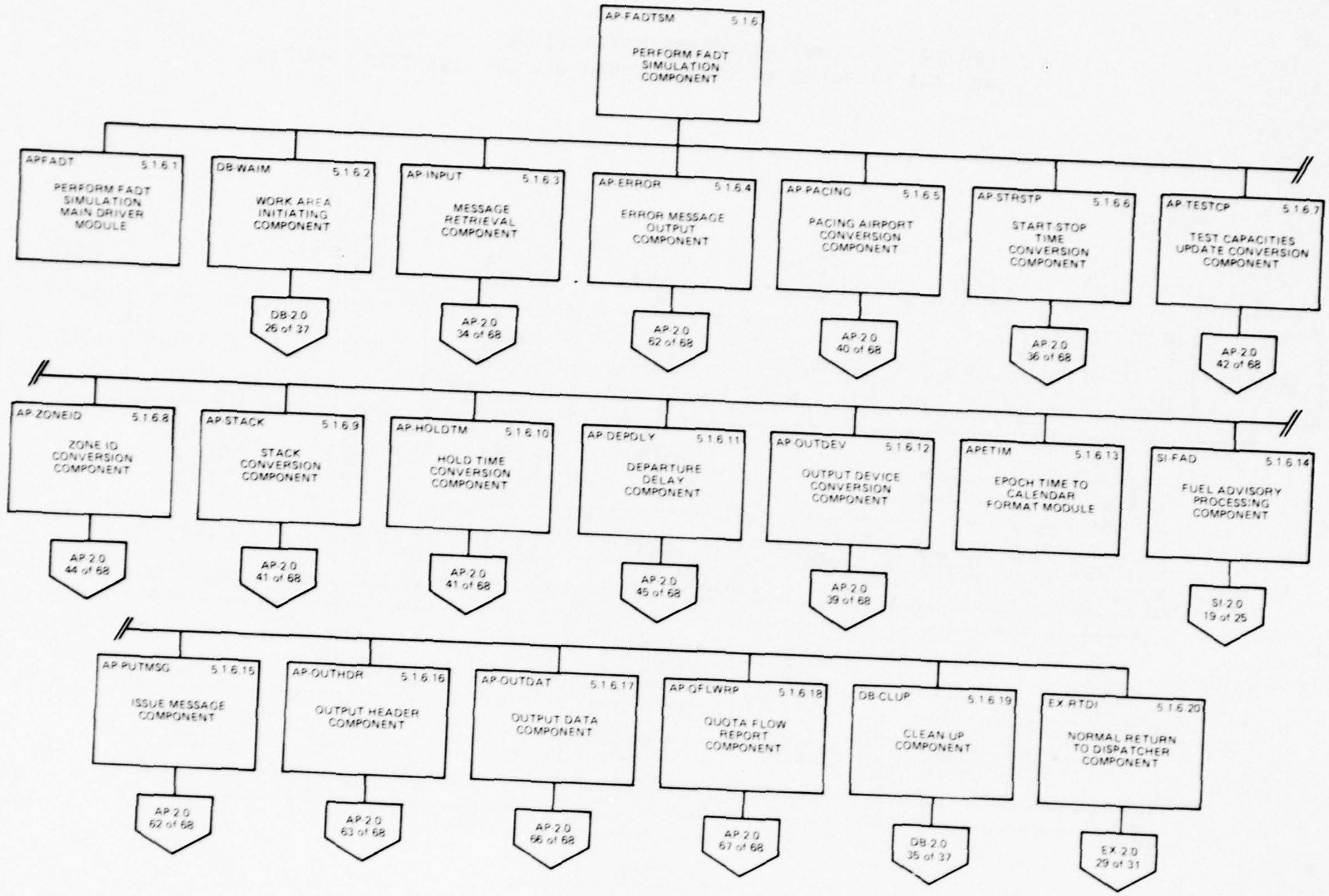


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (21 of 68)

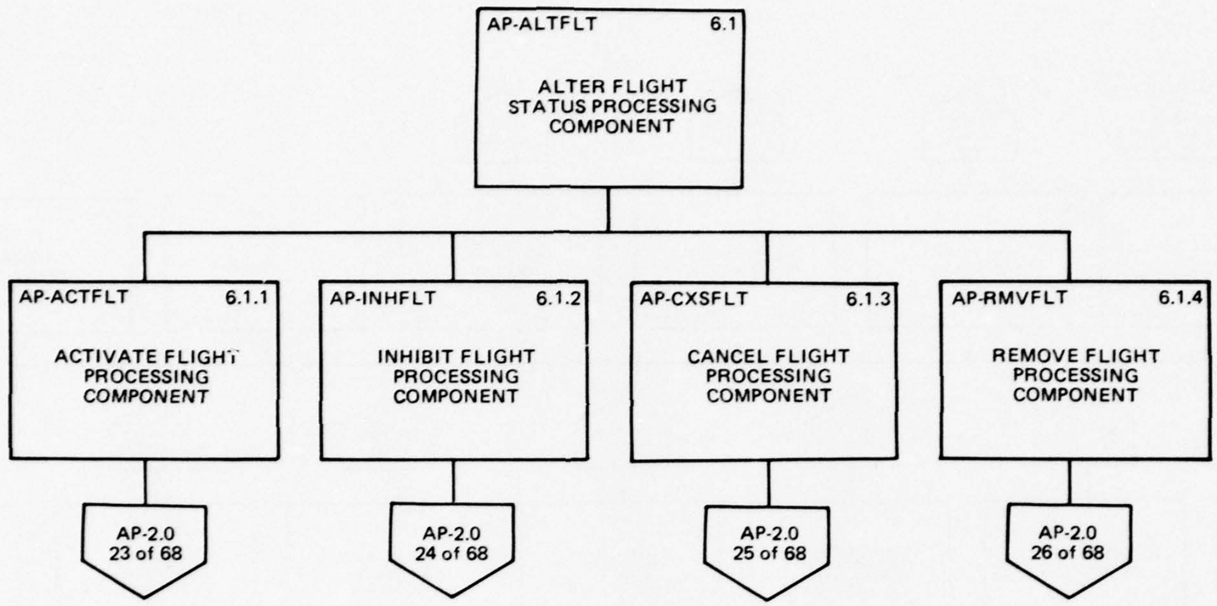


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (22 of 68)

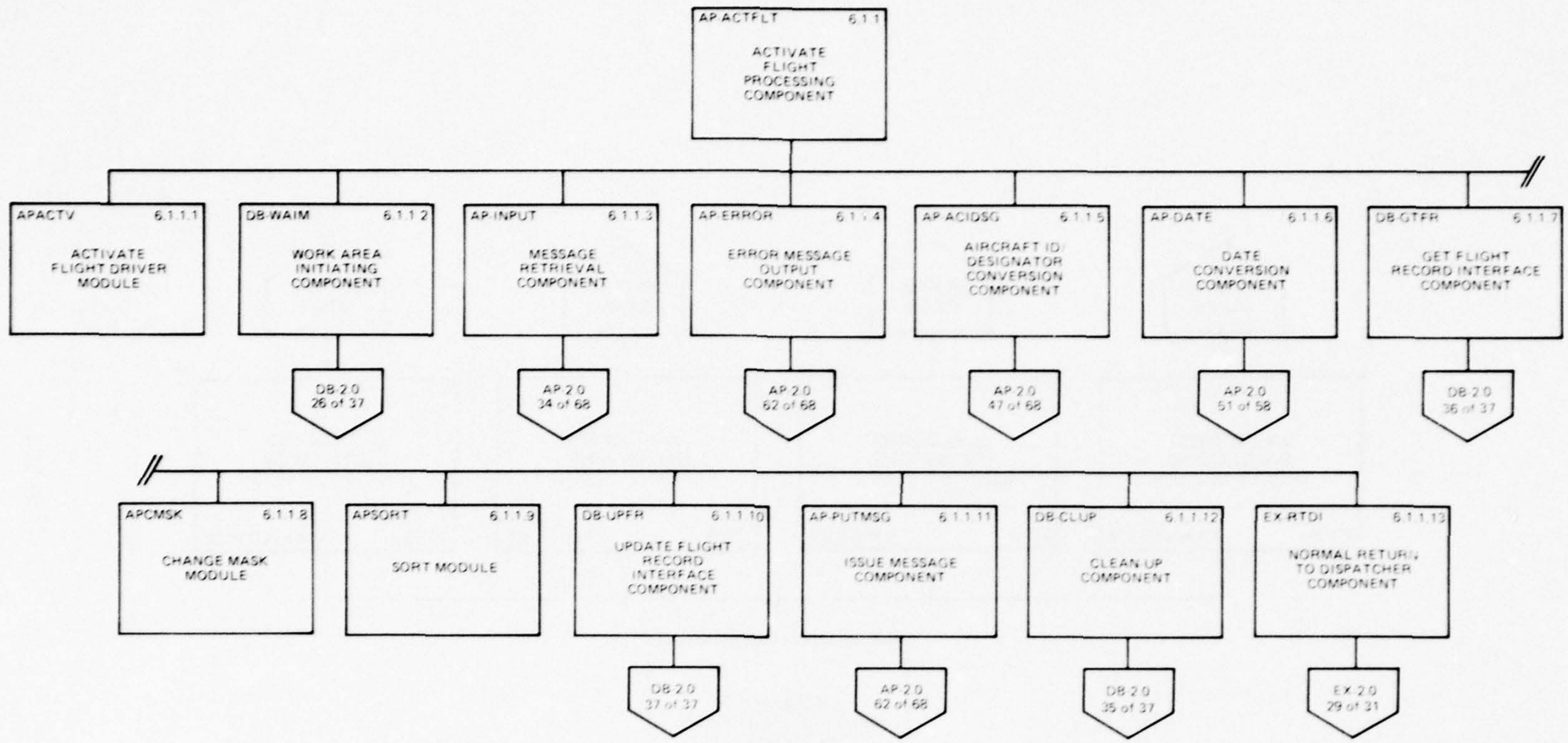


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (23 of 68)

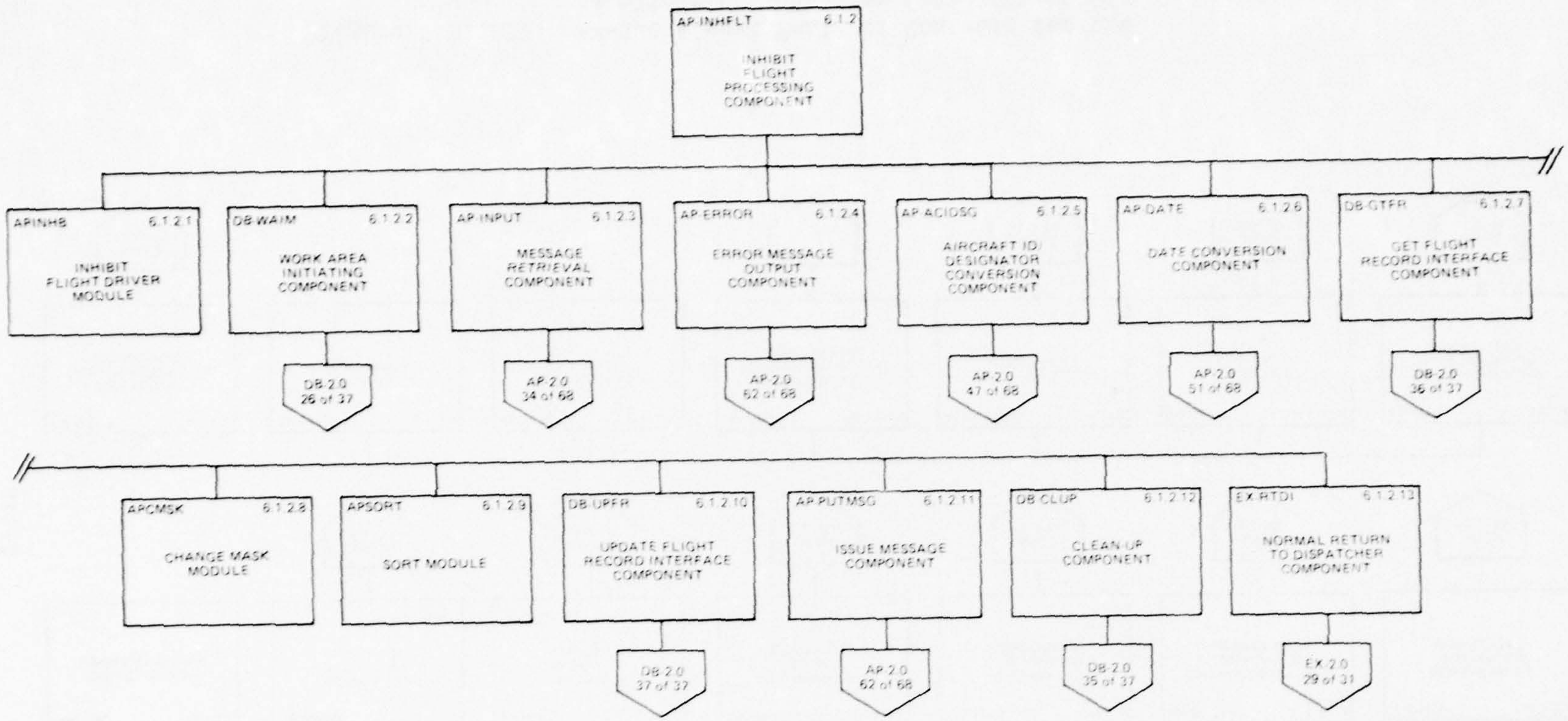


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (24 of 68)

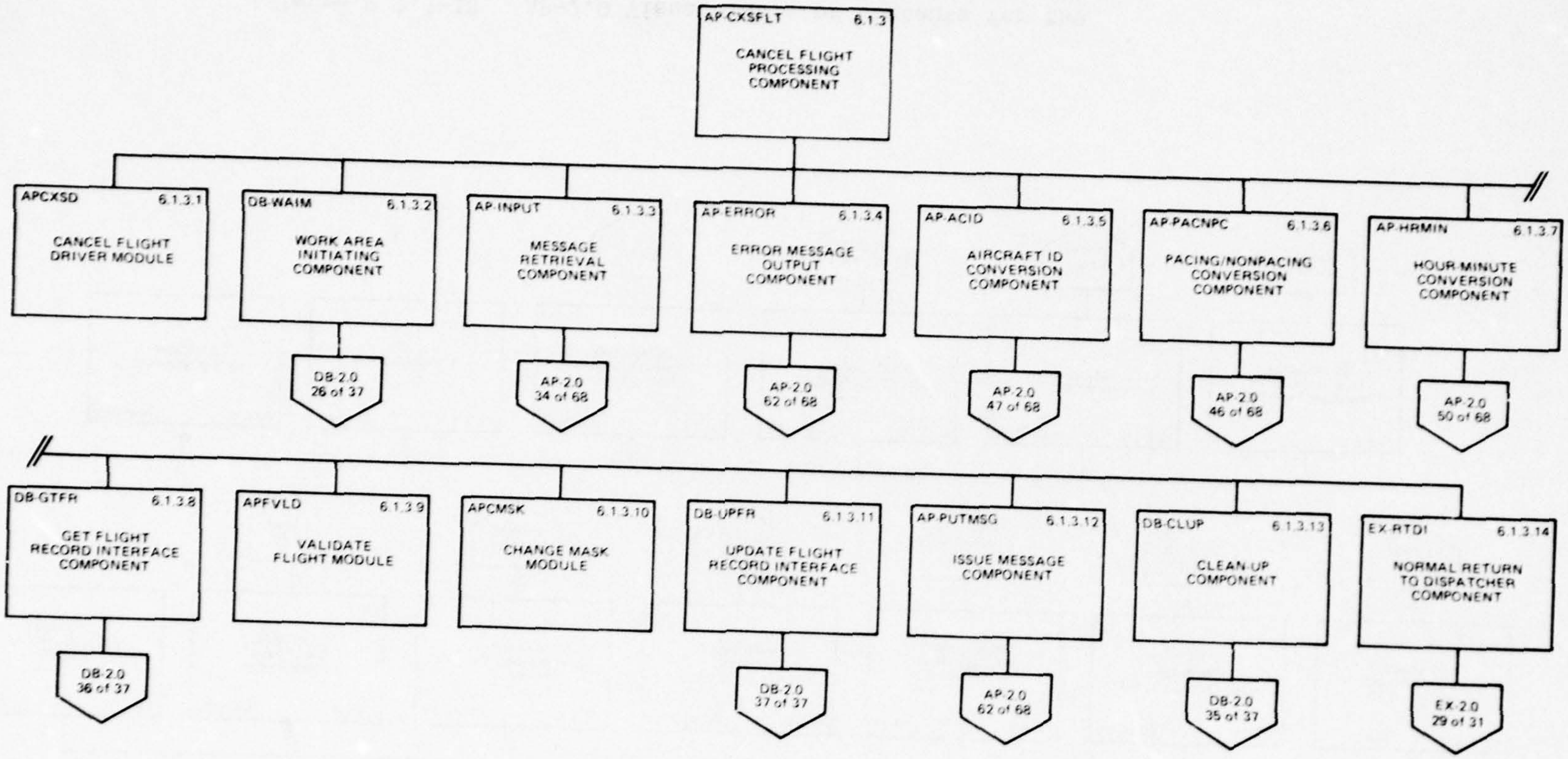


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (25 of 68)

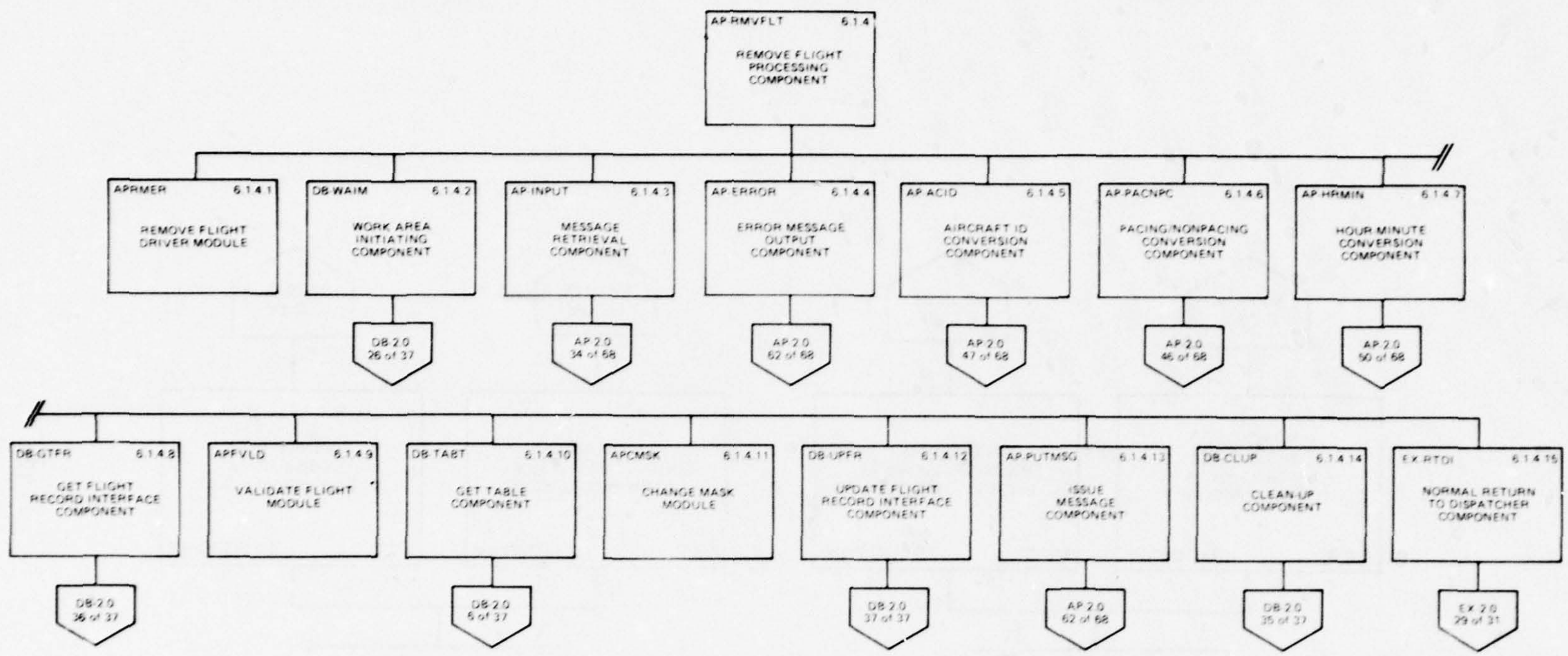


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (26 of 68)

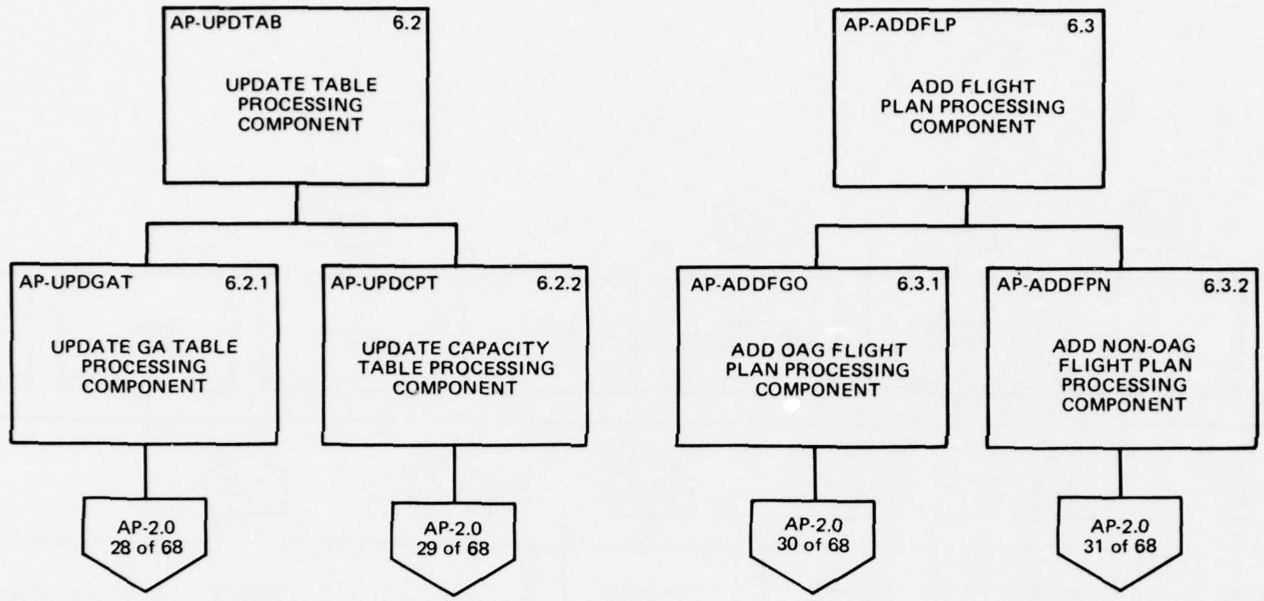


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (27 of 68)

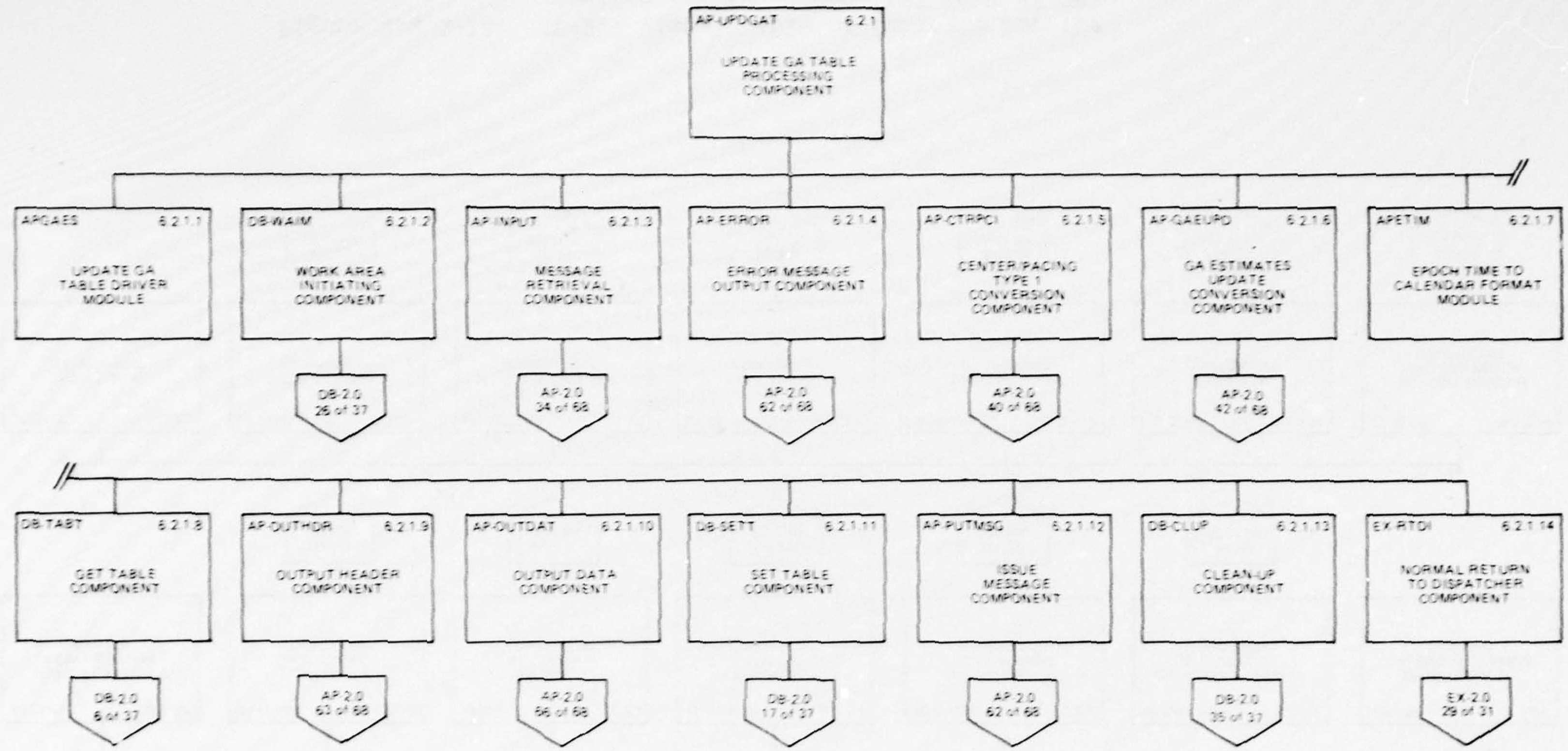


Figure 2.2.1-13. *AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (28 of 68)*

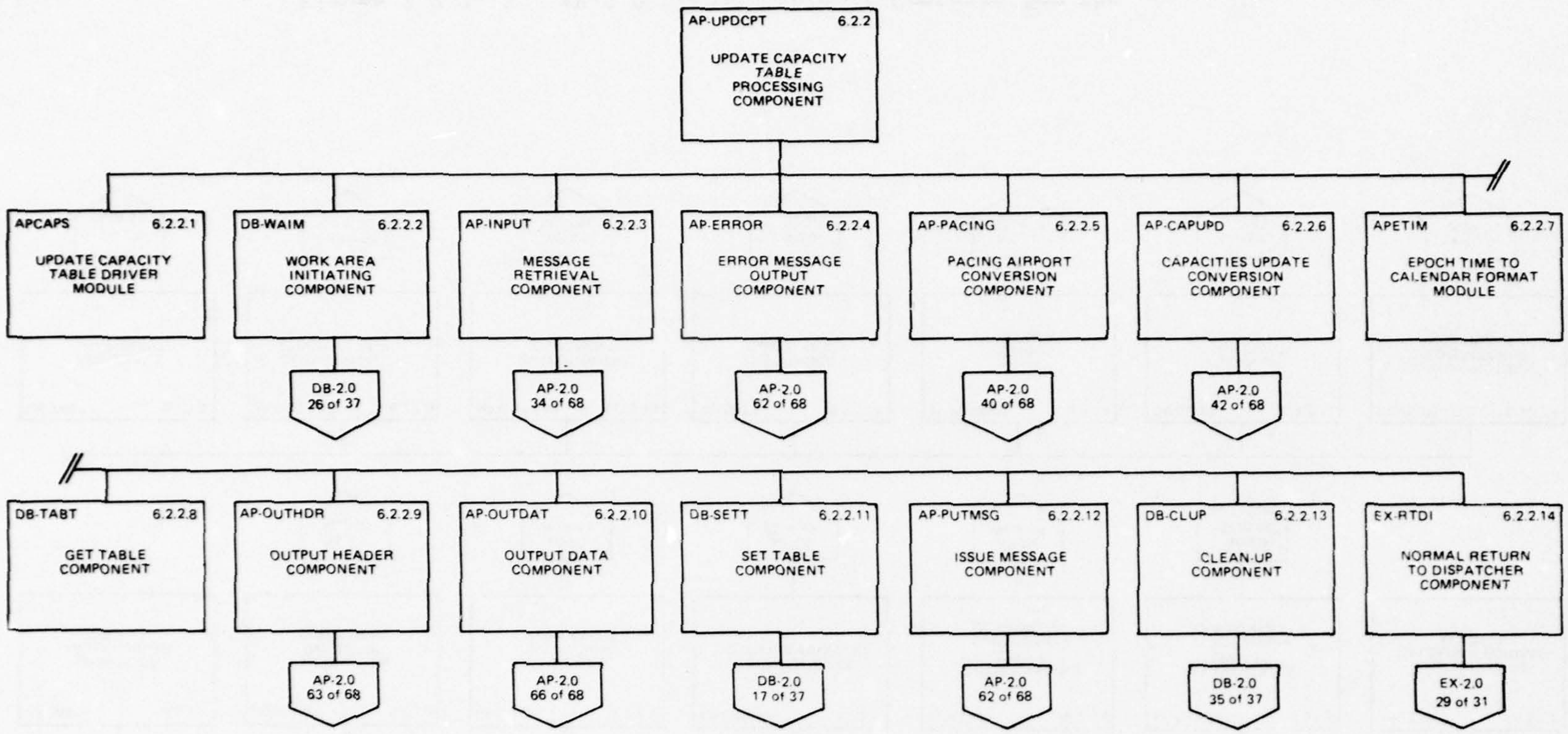


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (29 of 68)

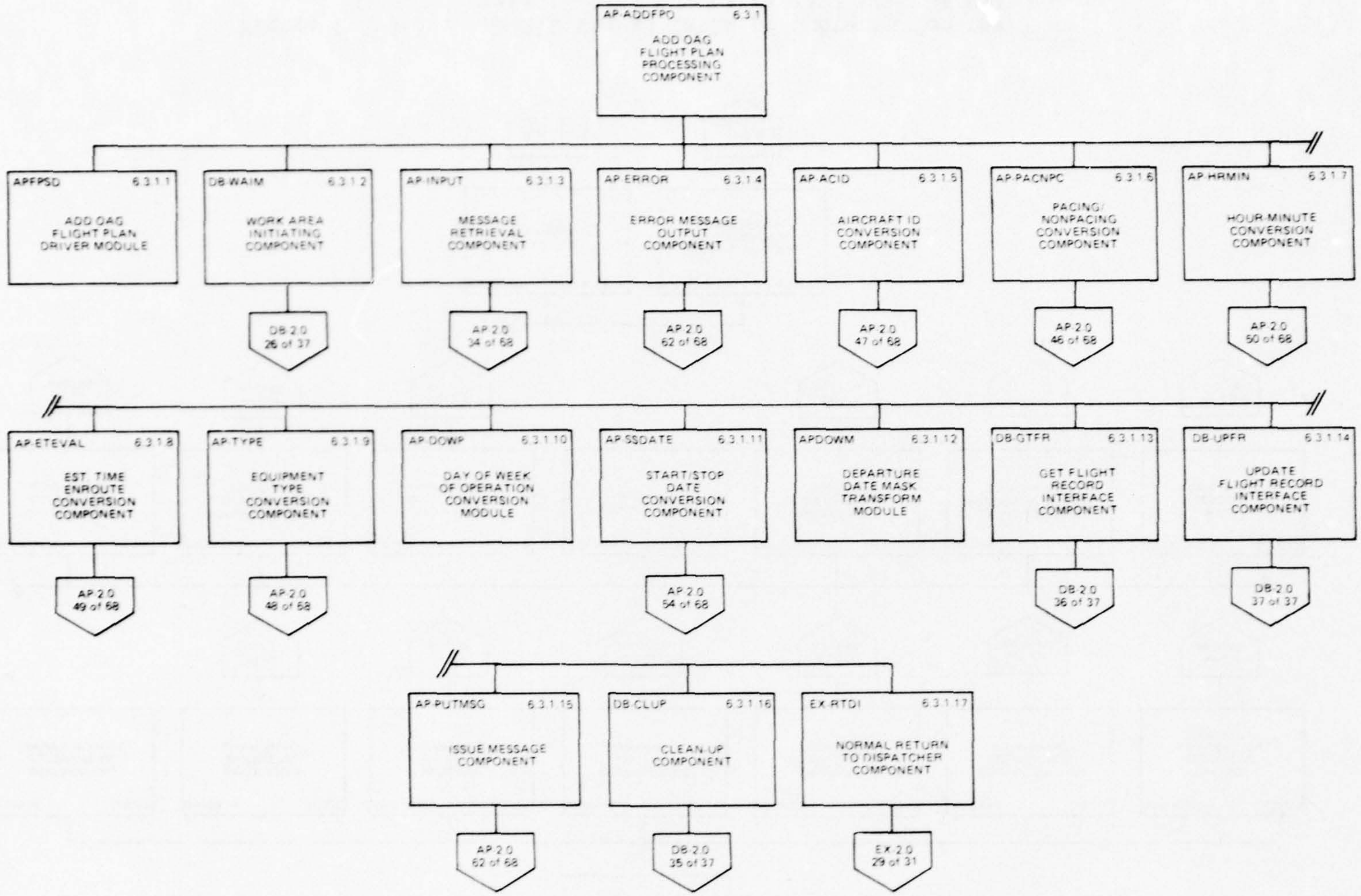


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (30 of 68)

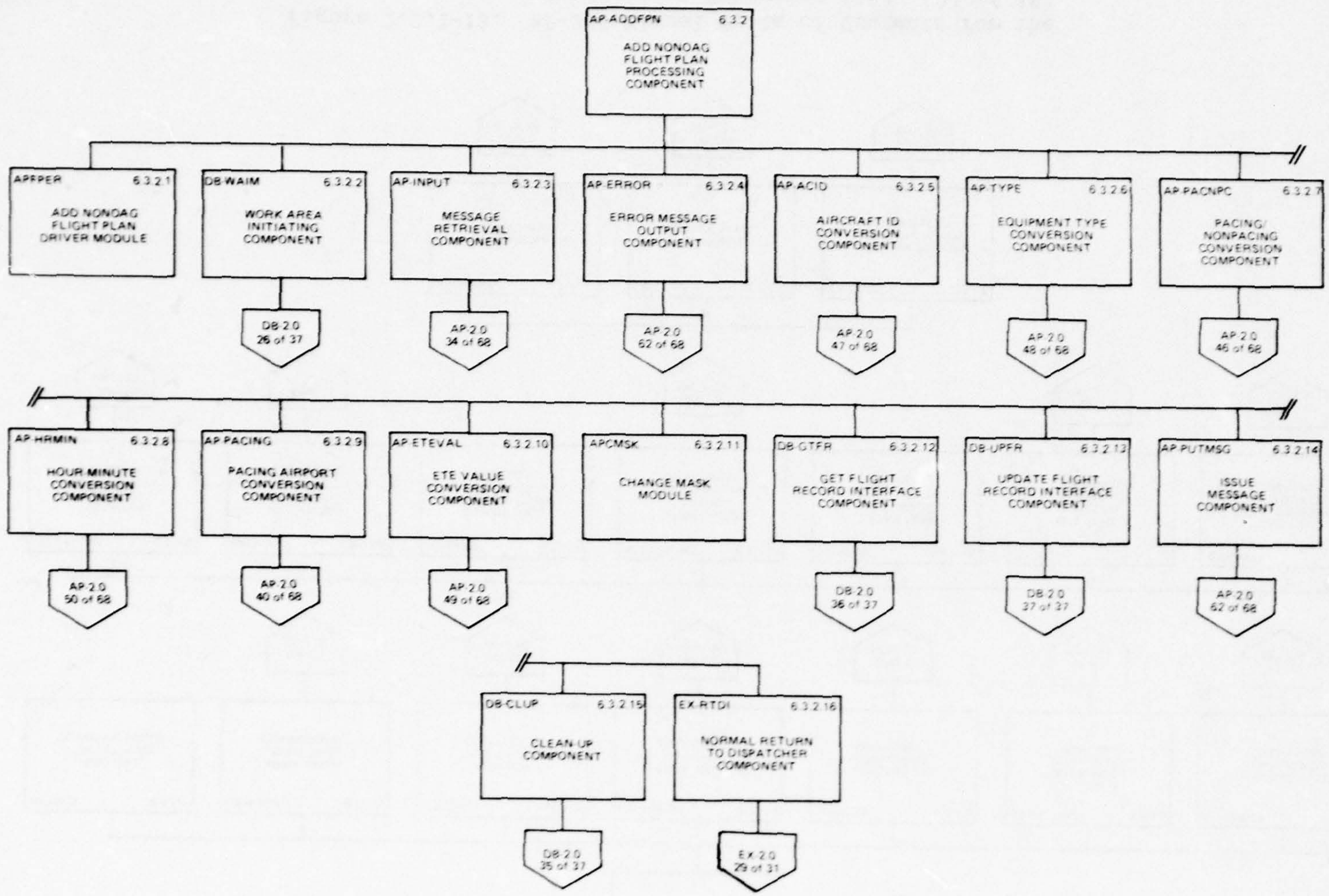


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (31 of 68)

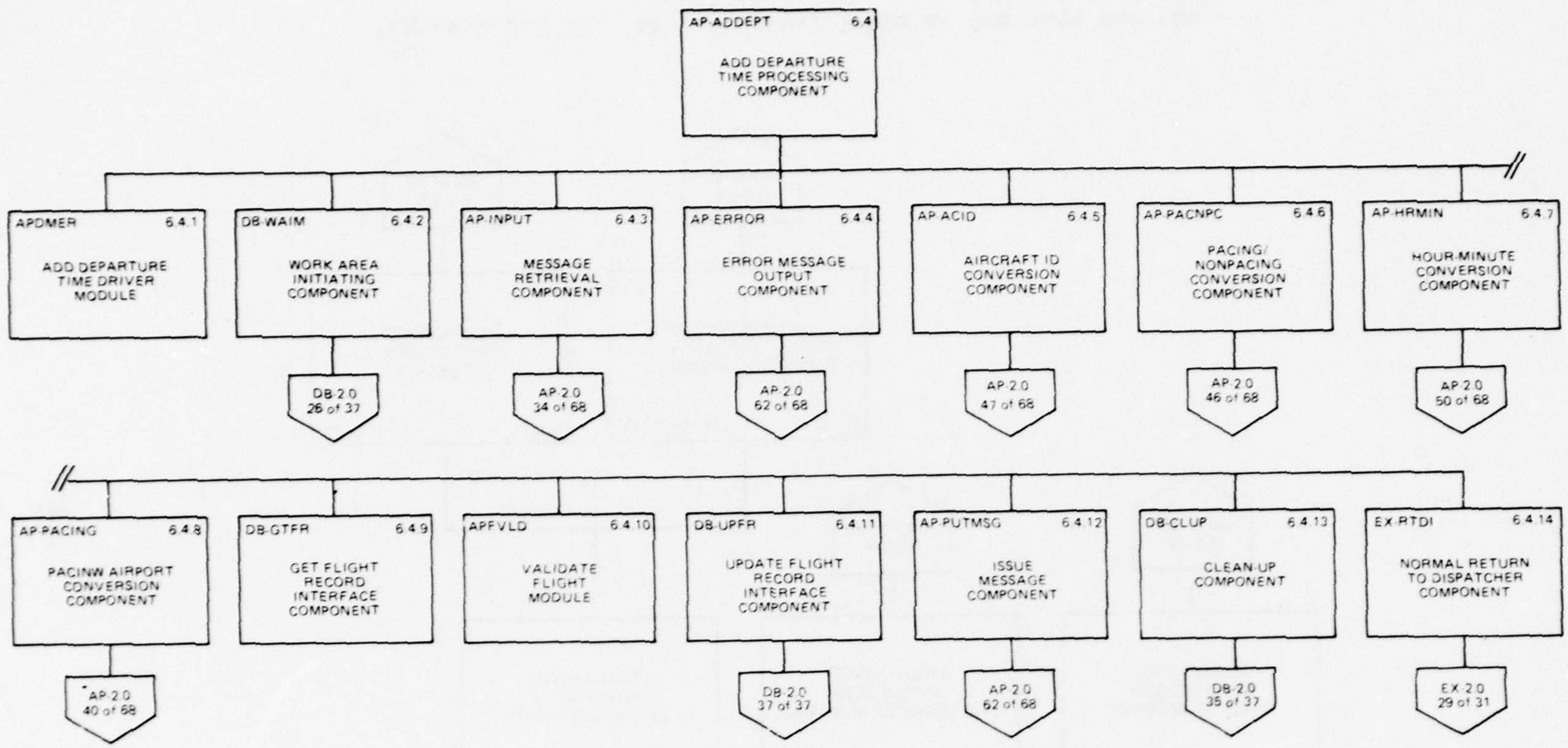


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (32 of 68)

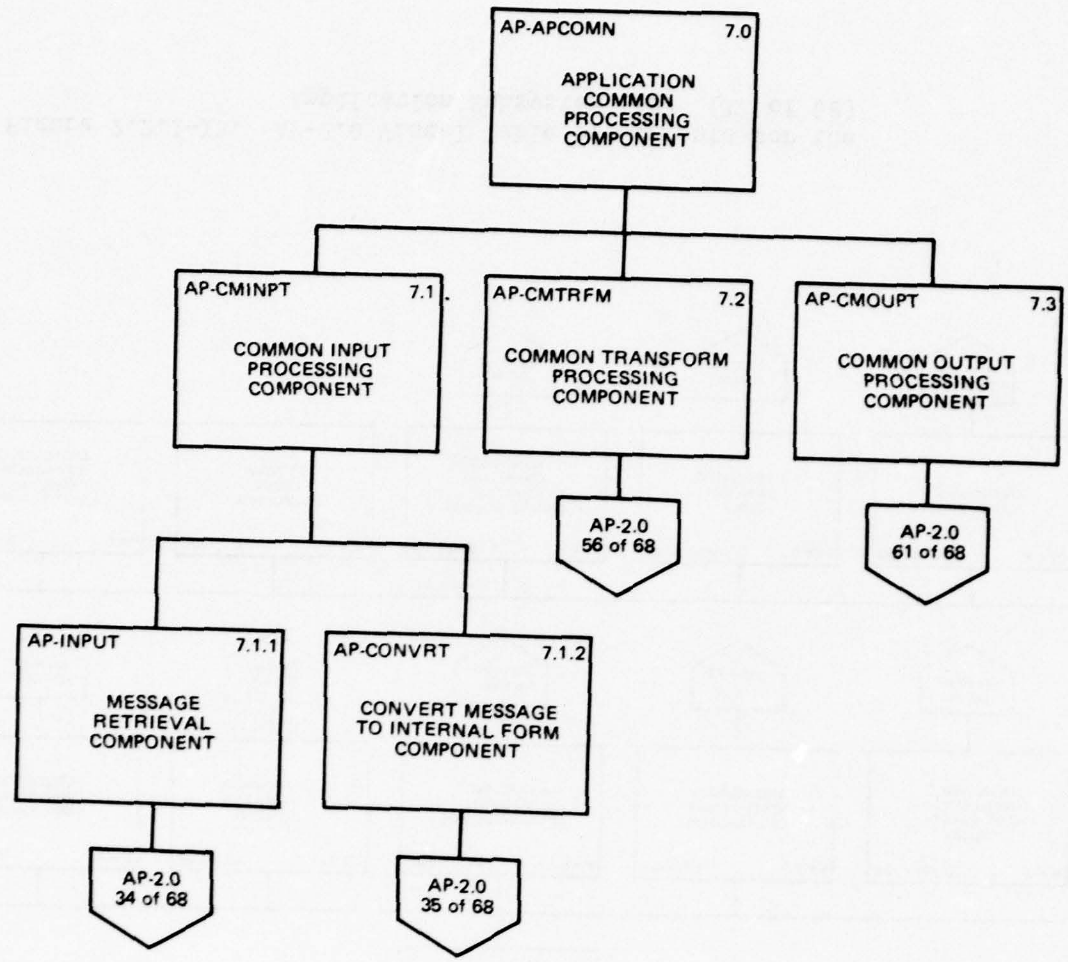


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (33 of 68)

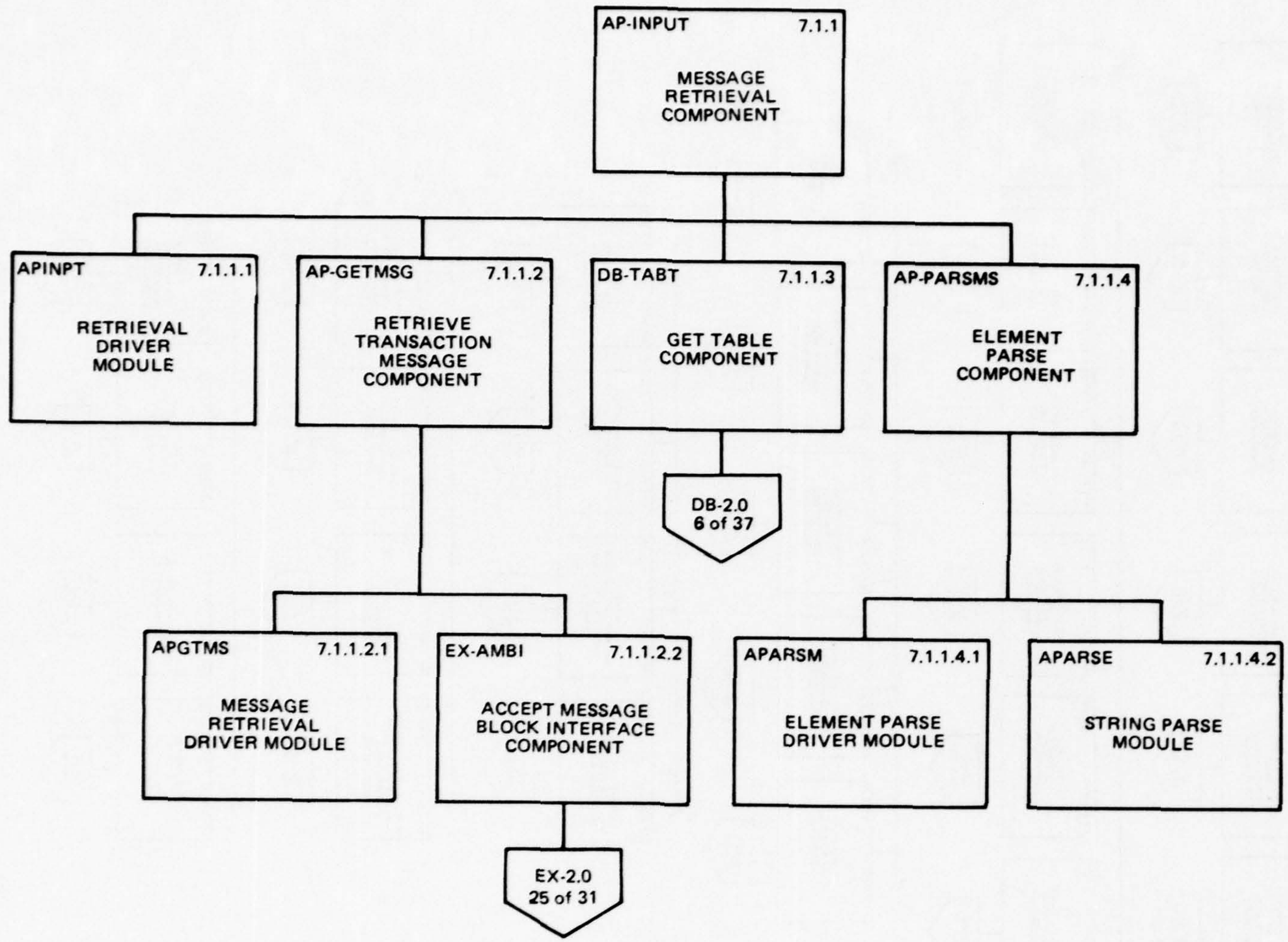


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (34 of 68)

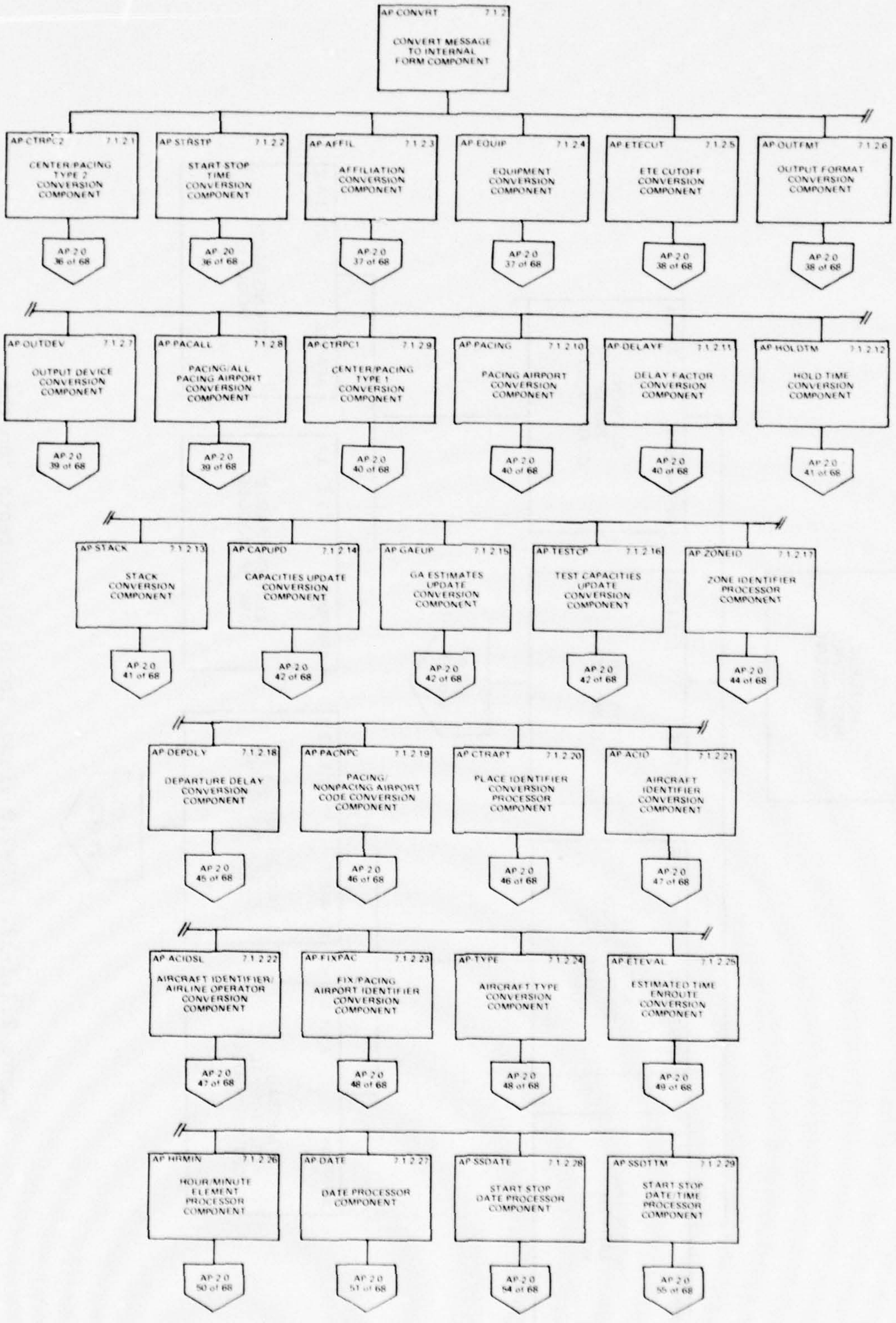


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (35 of 68)

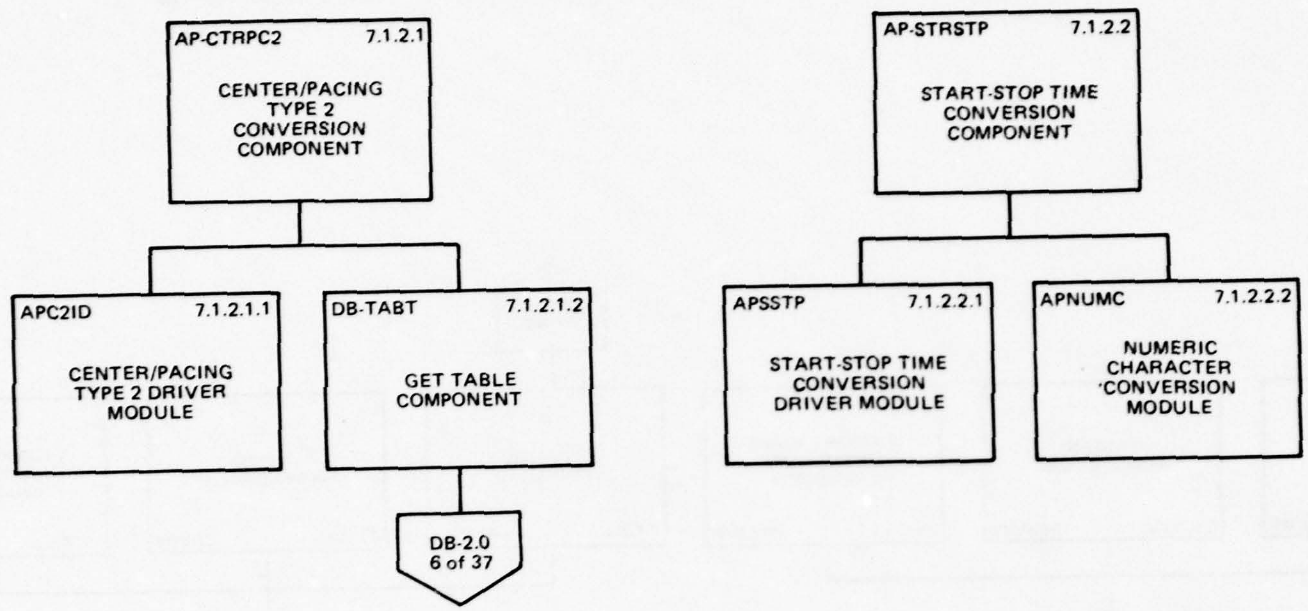


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (36 of 68)

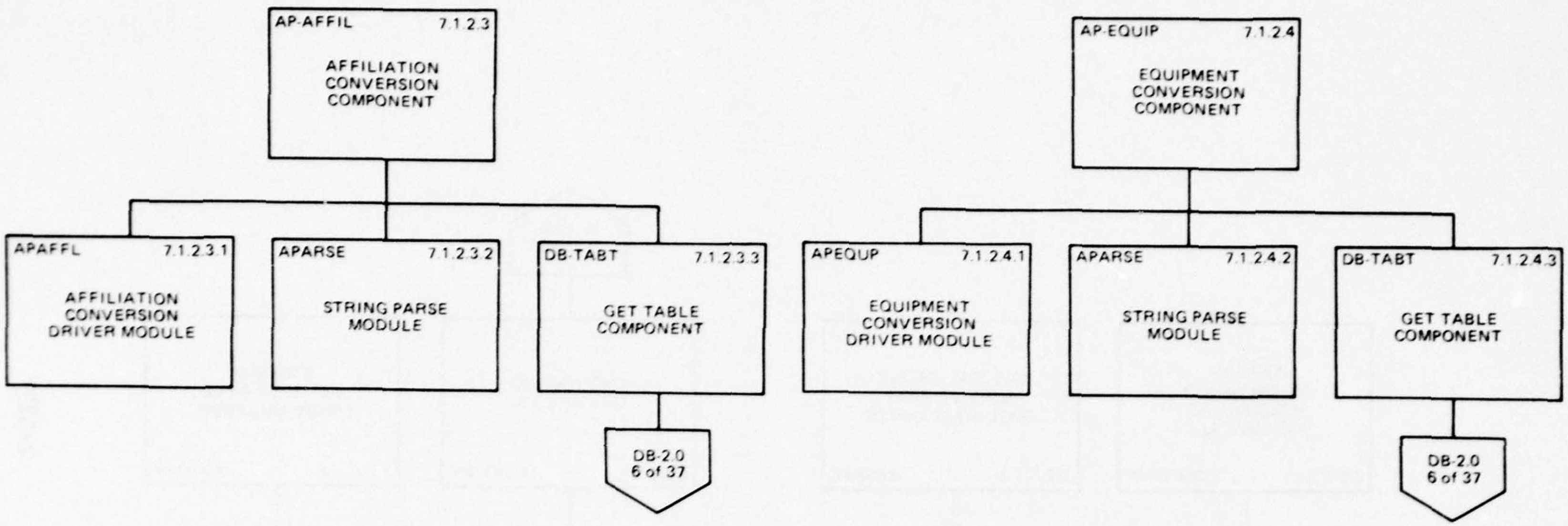


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (37 of 68)

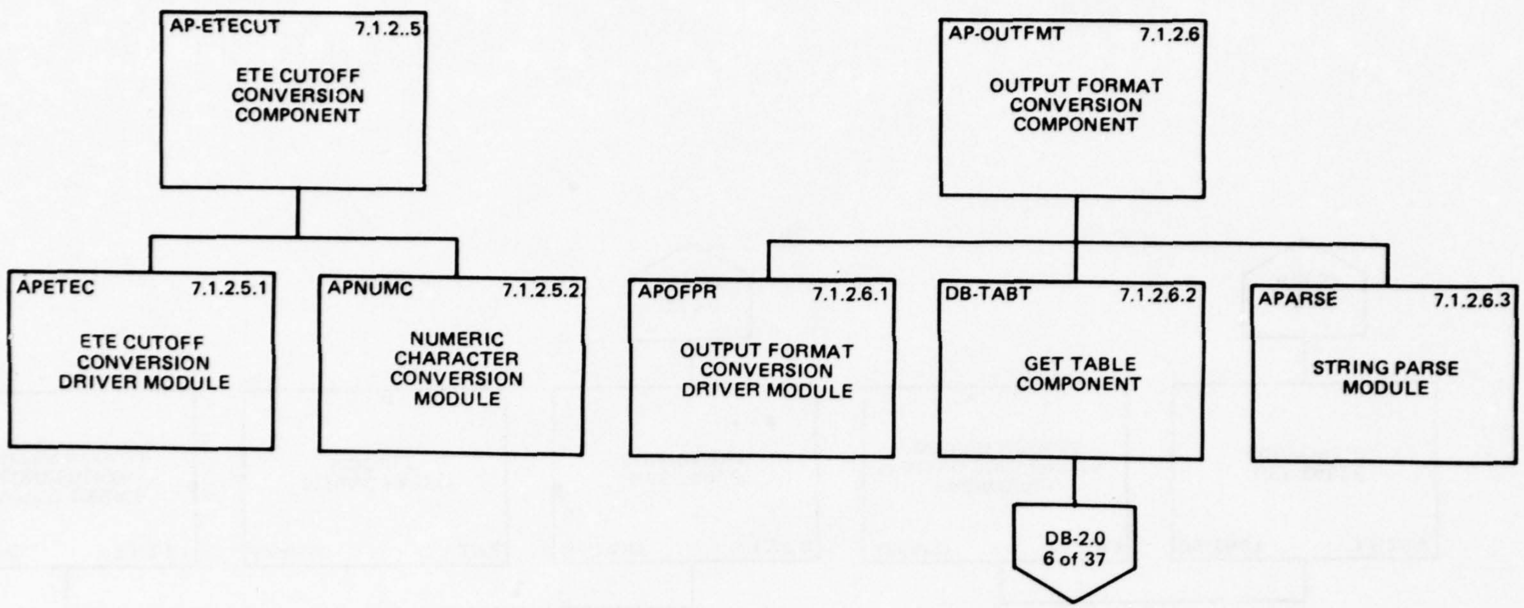


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (38 of 68)

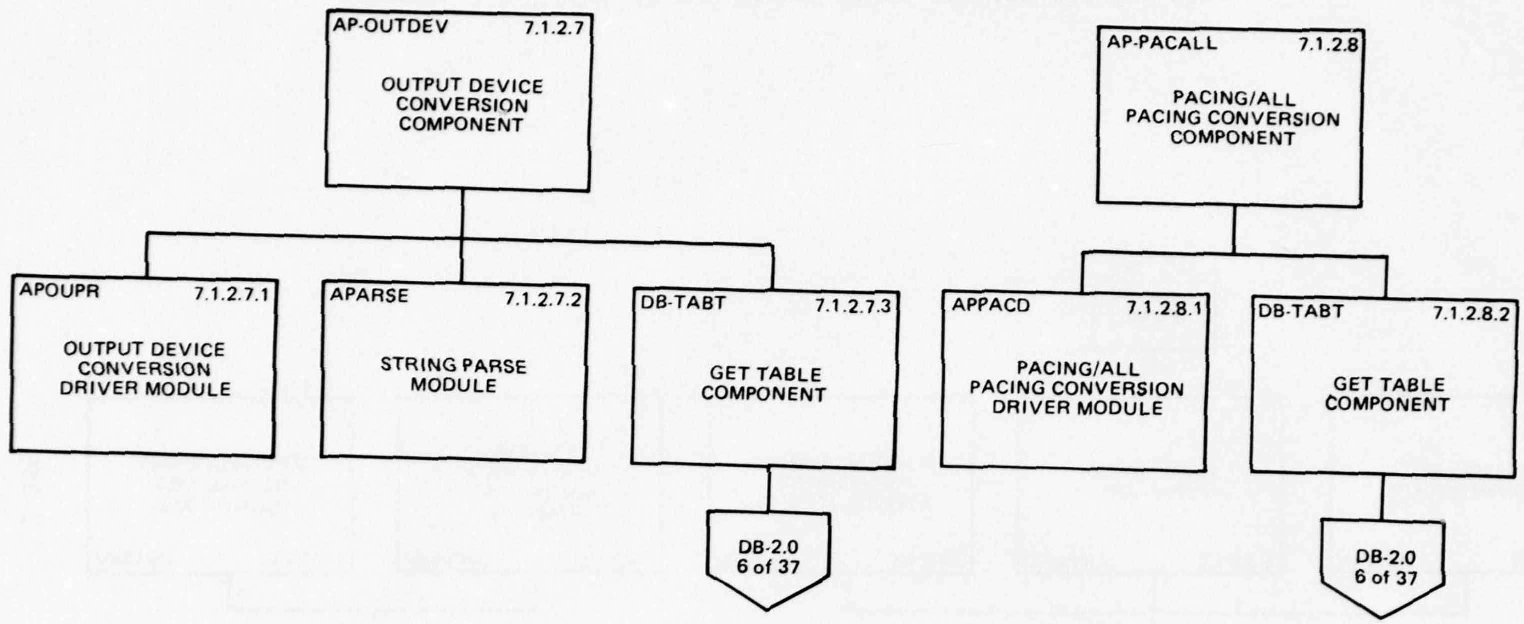


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (39 of 68)

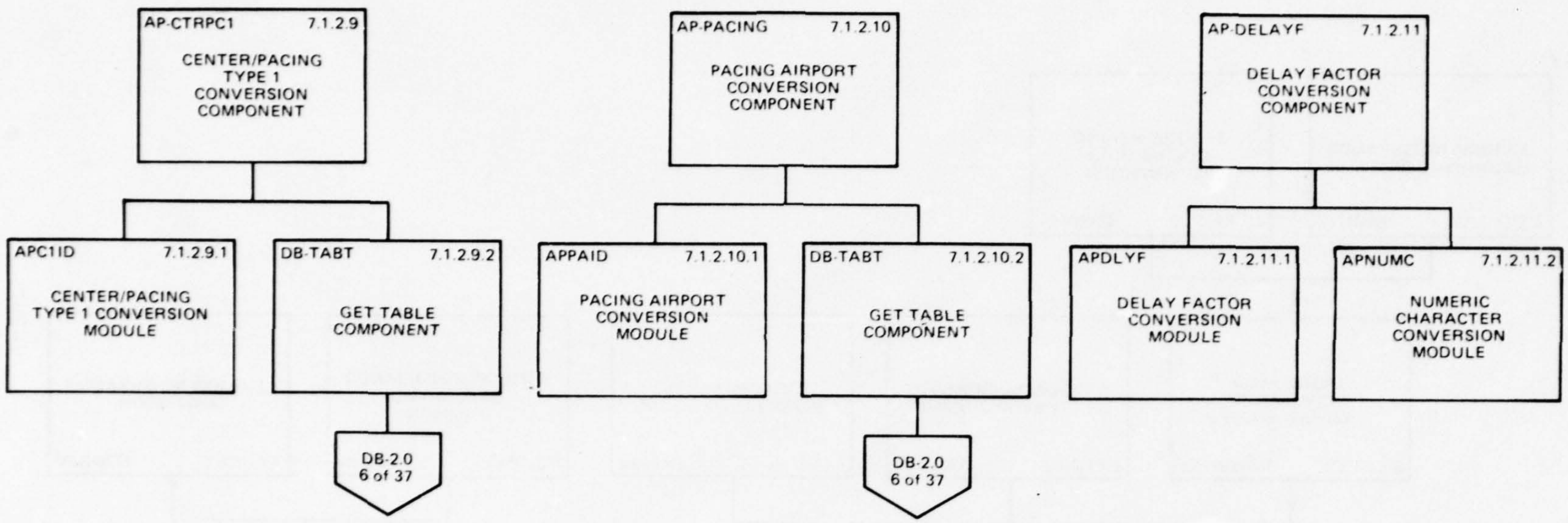


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (40 of 68)

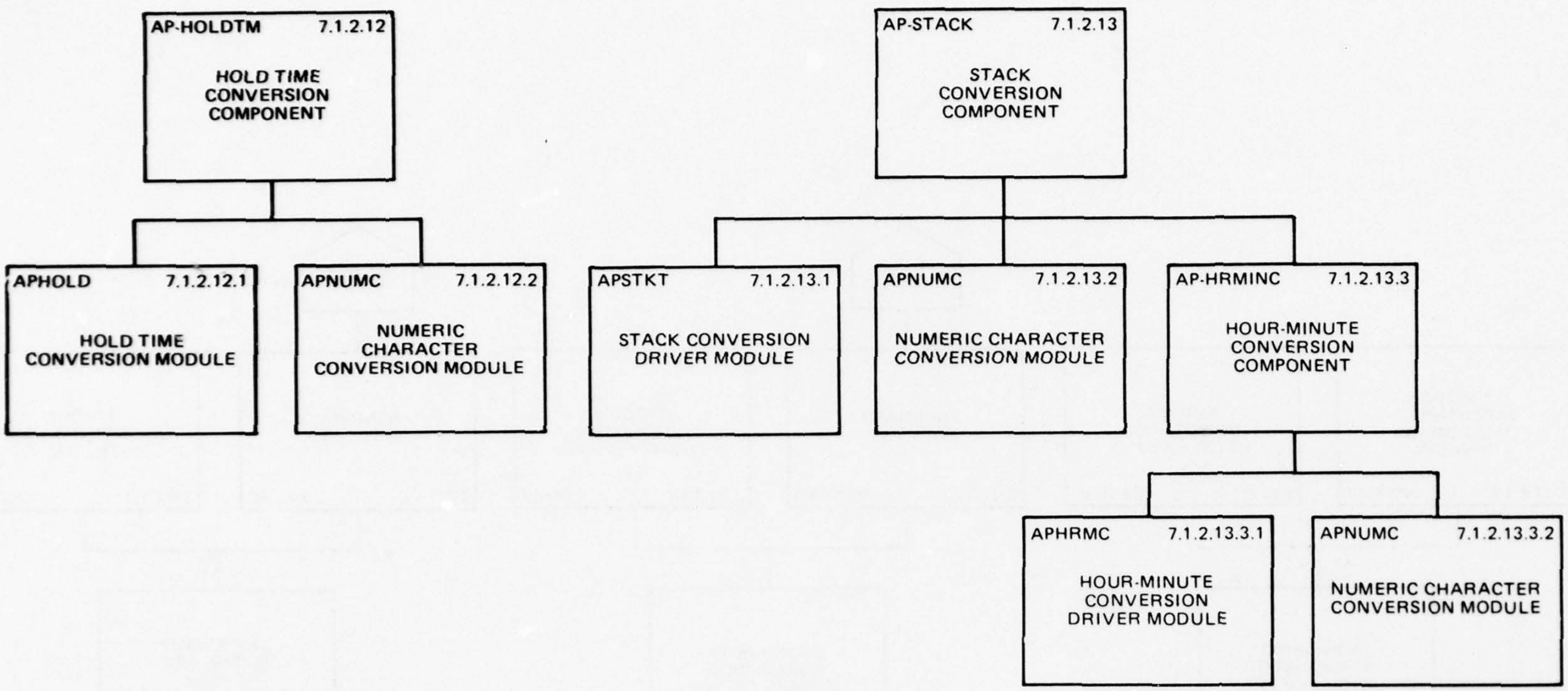


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (41 of 68)

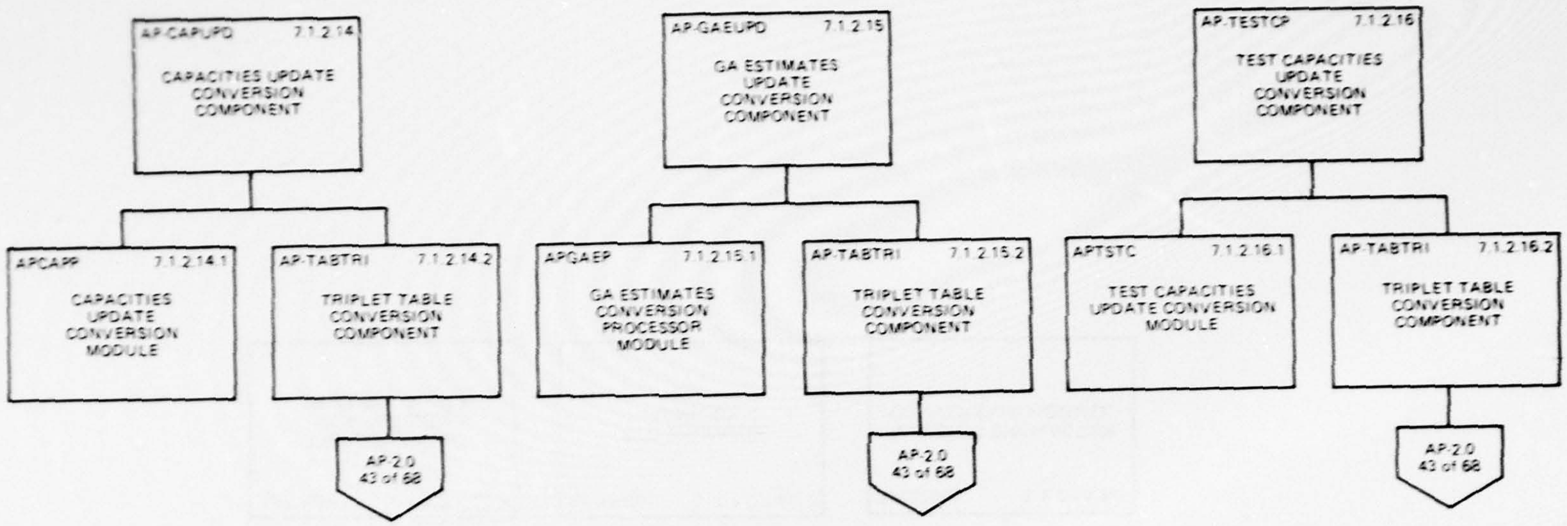


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (42 of 68)

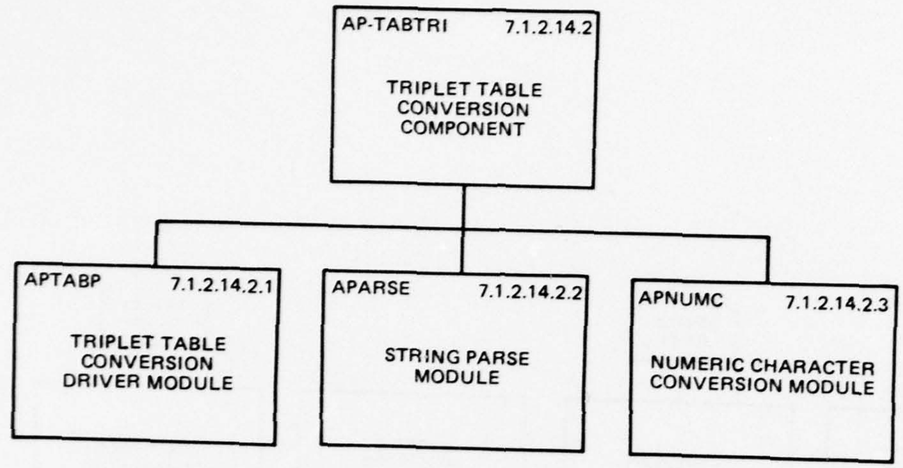


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (43 of 68)

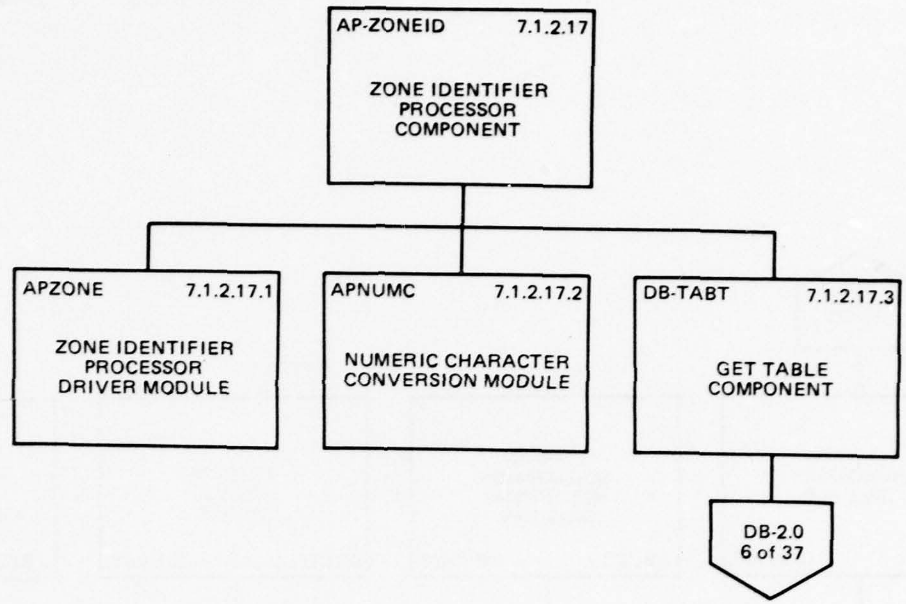


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (44 of 68)

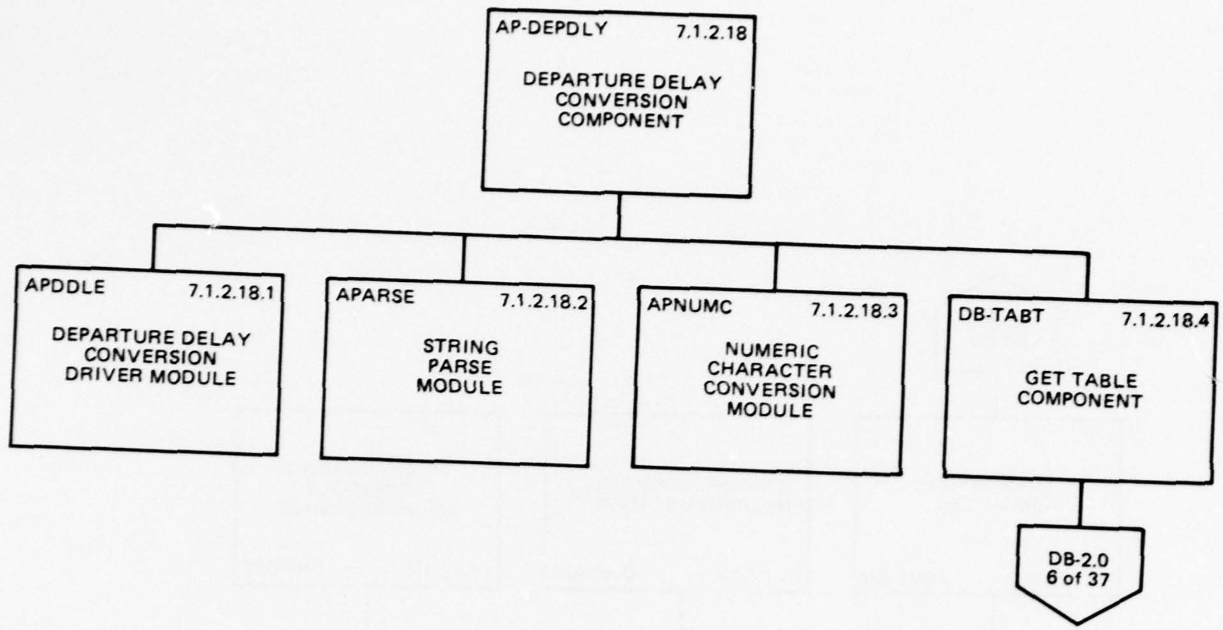


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (45 of 68)

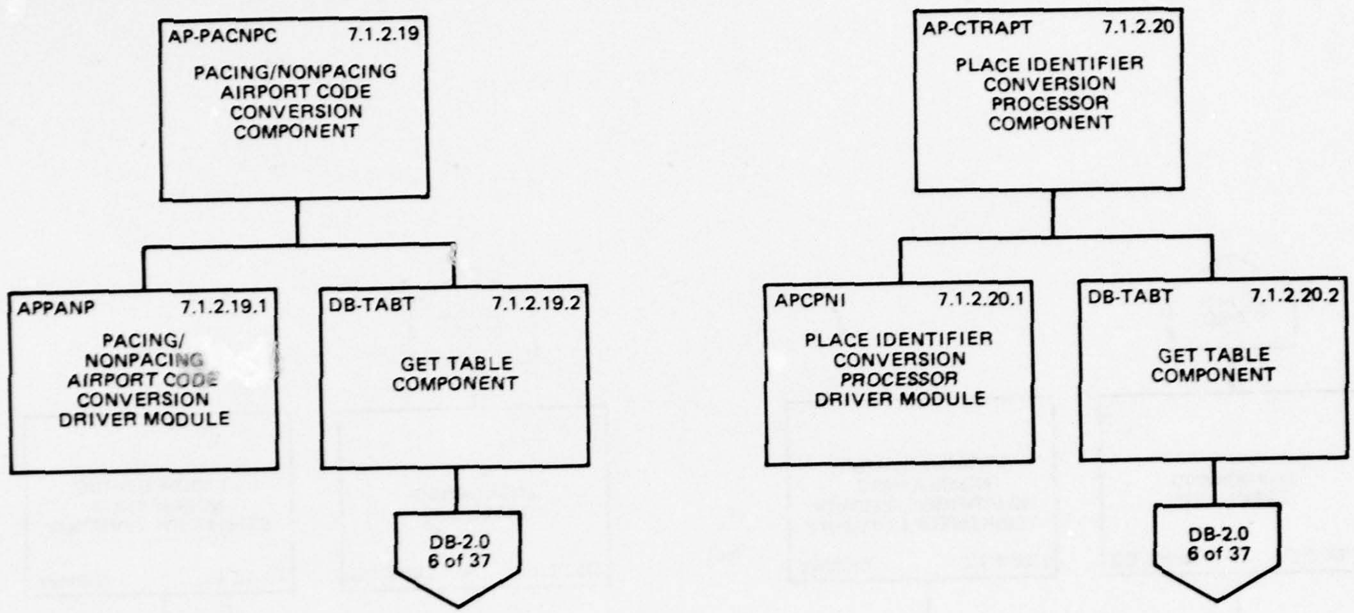


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (46 of 68)

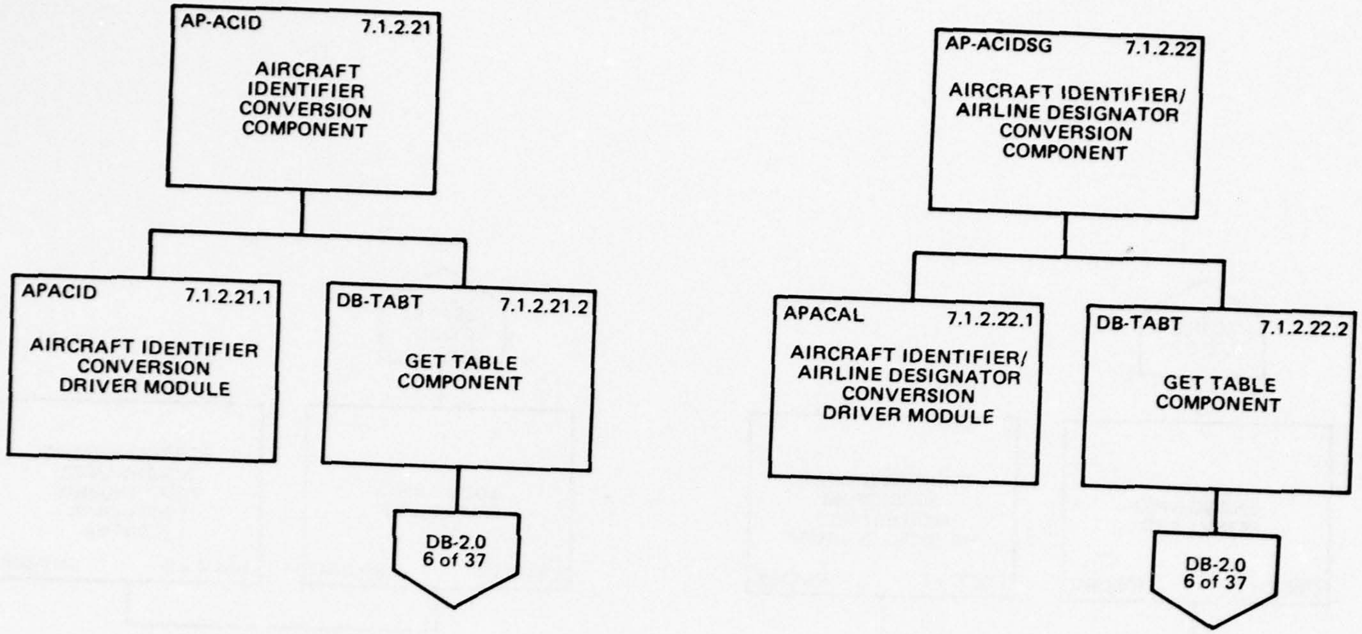


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (47 of 68)

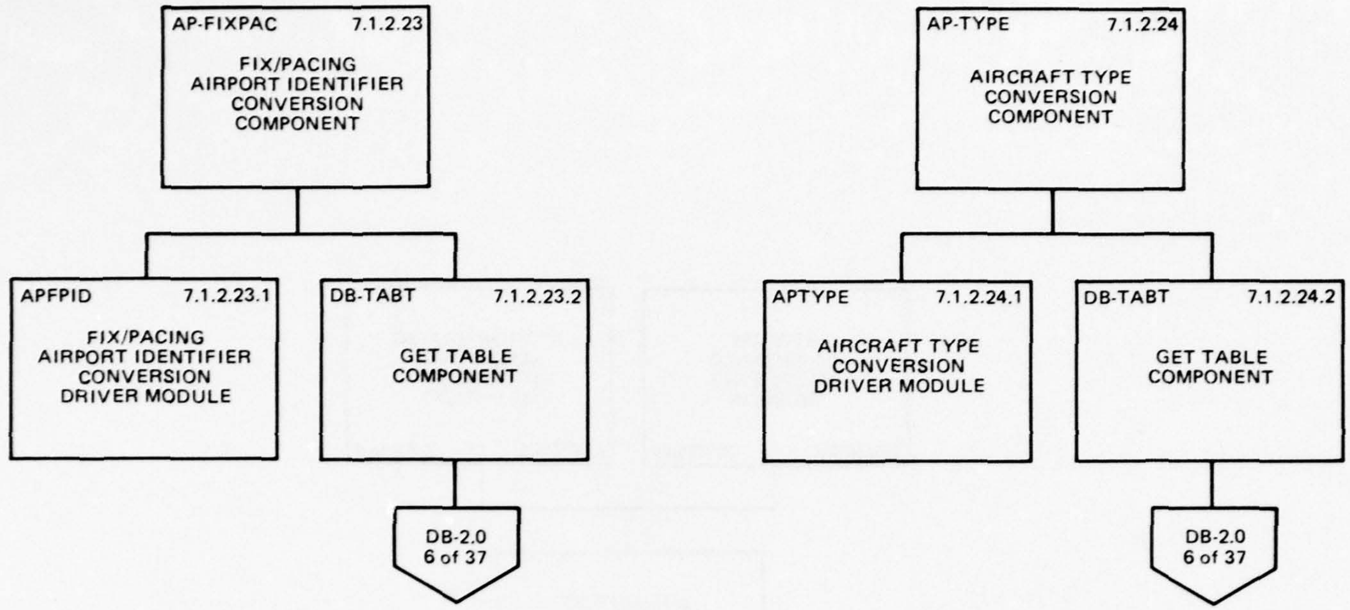


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (48 of 68)

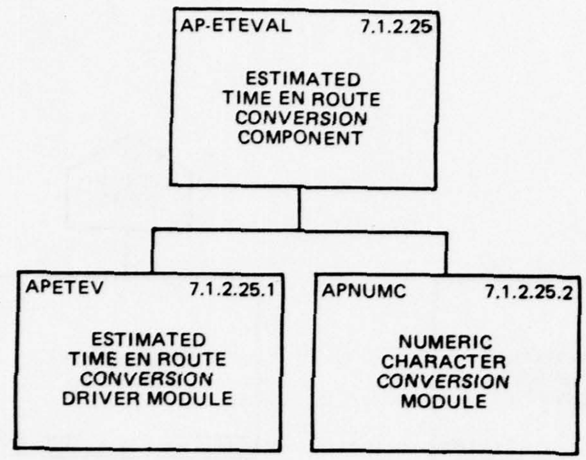


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (49 of 68)

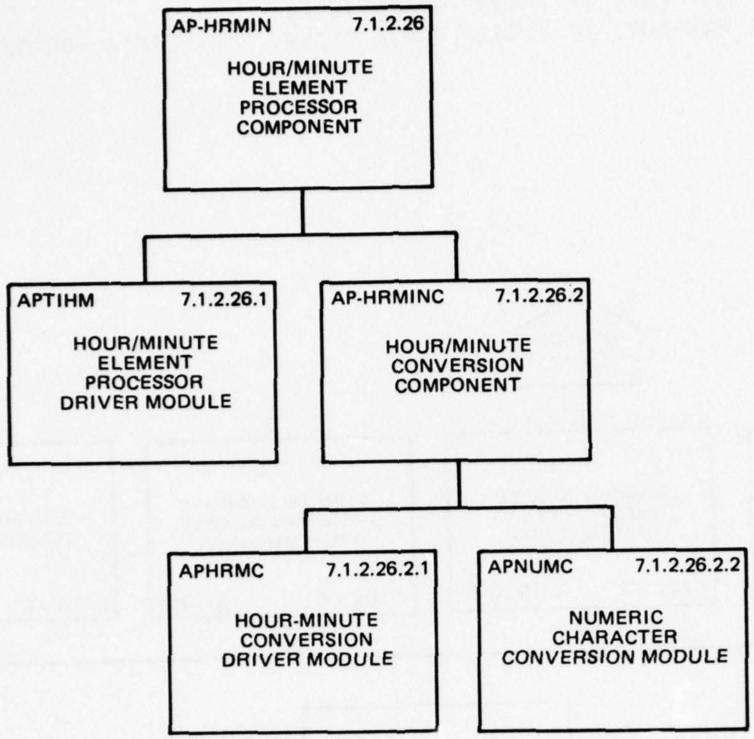


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (50 of 68)

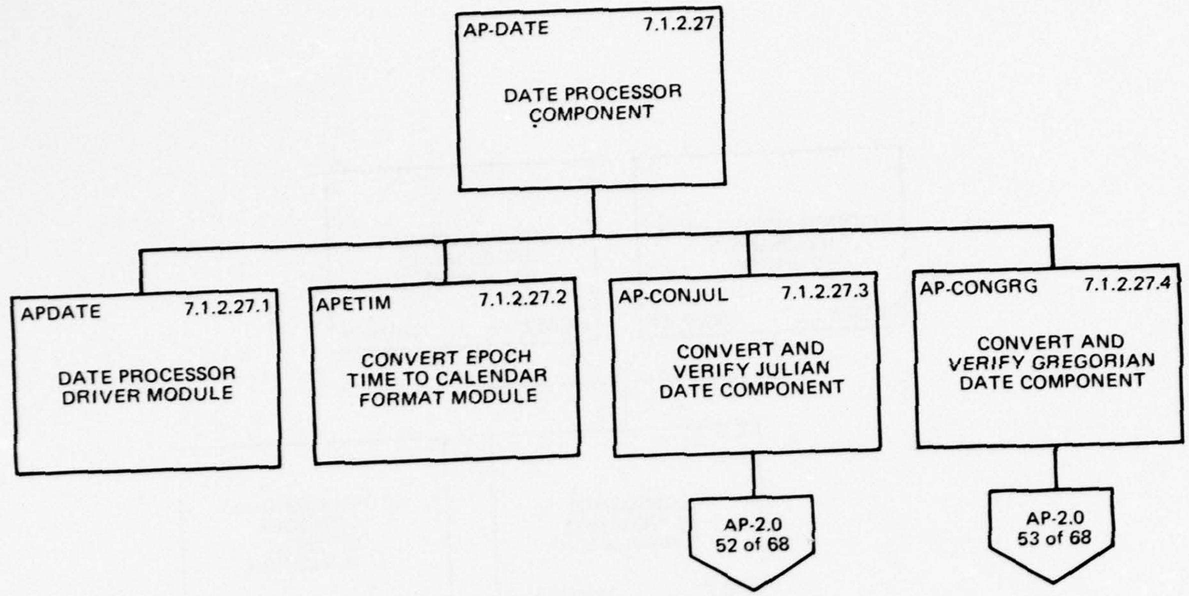


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (51 of 68)

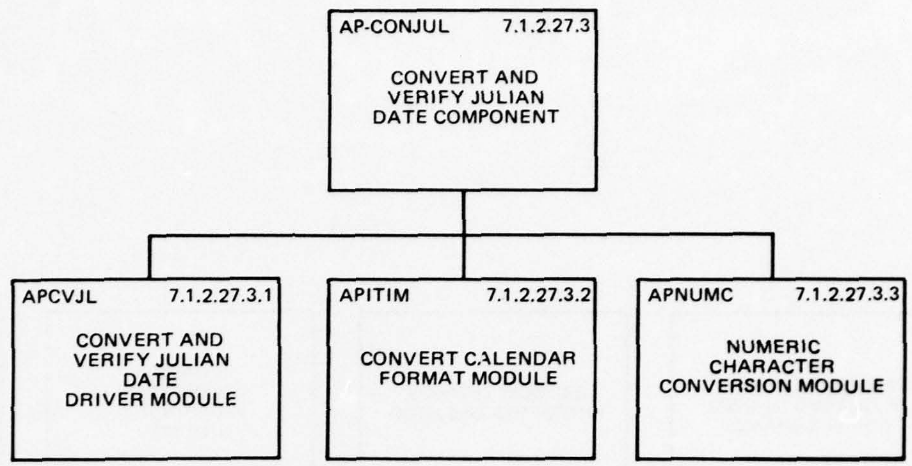


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (52 of 68)

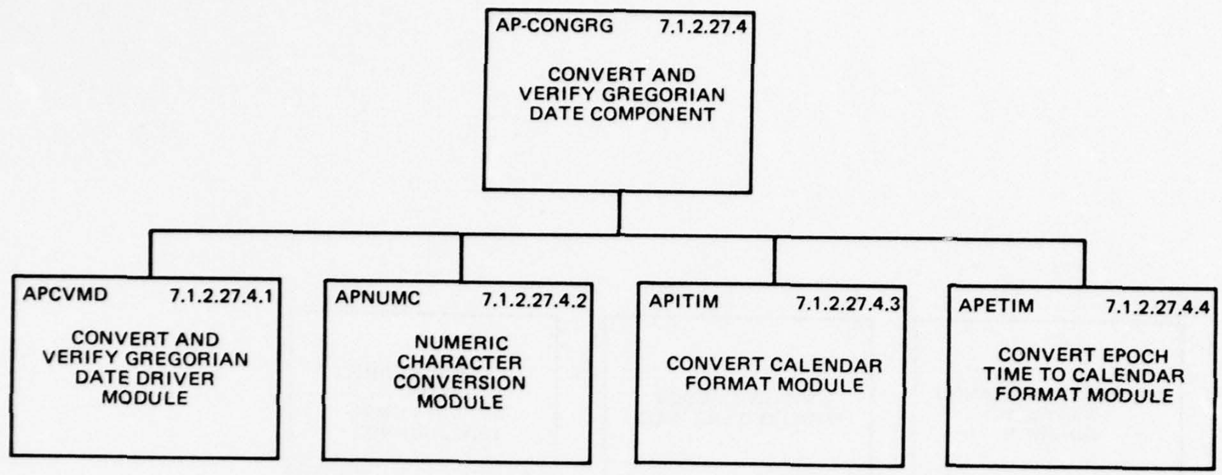


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (53 of 68)

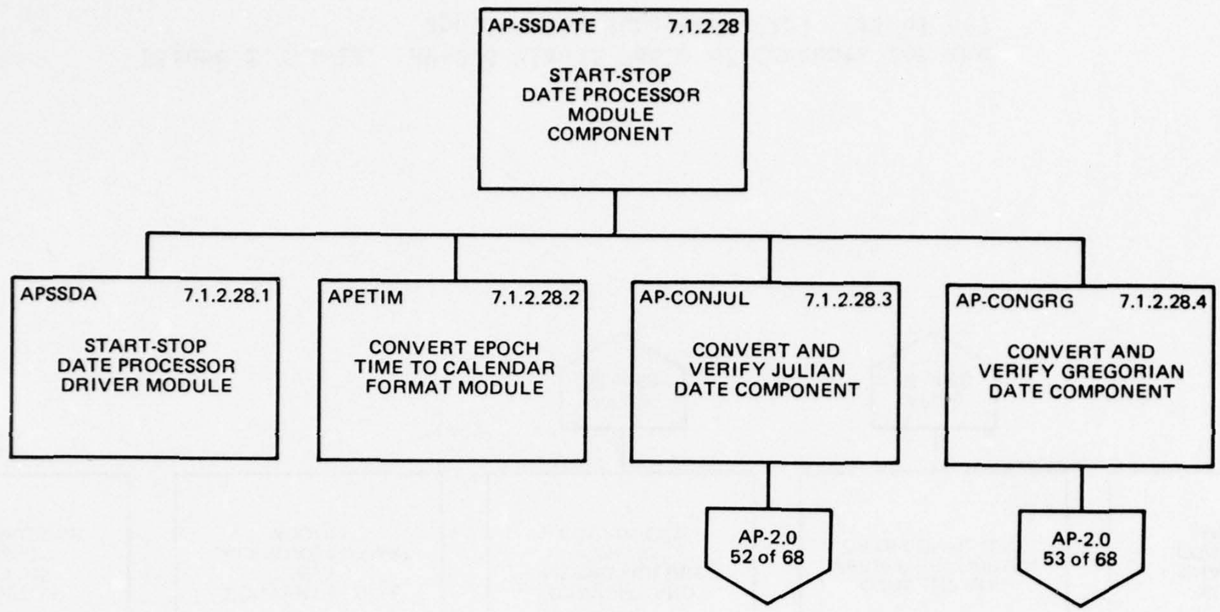


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (54 of 68)

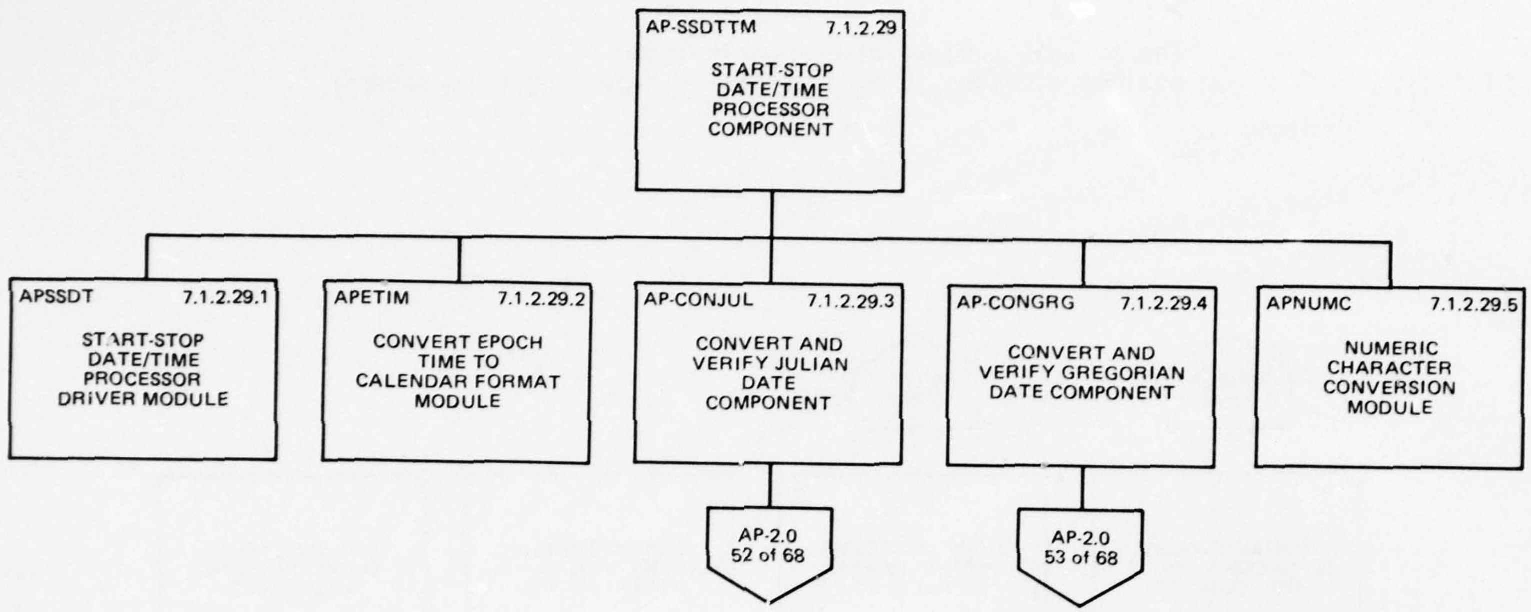
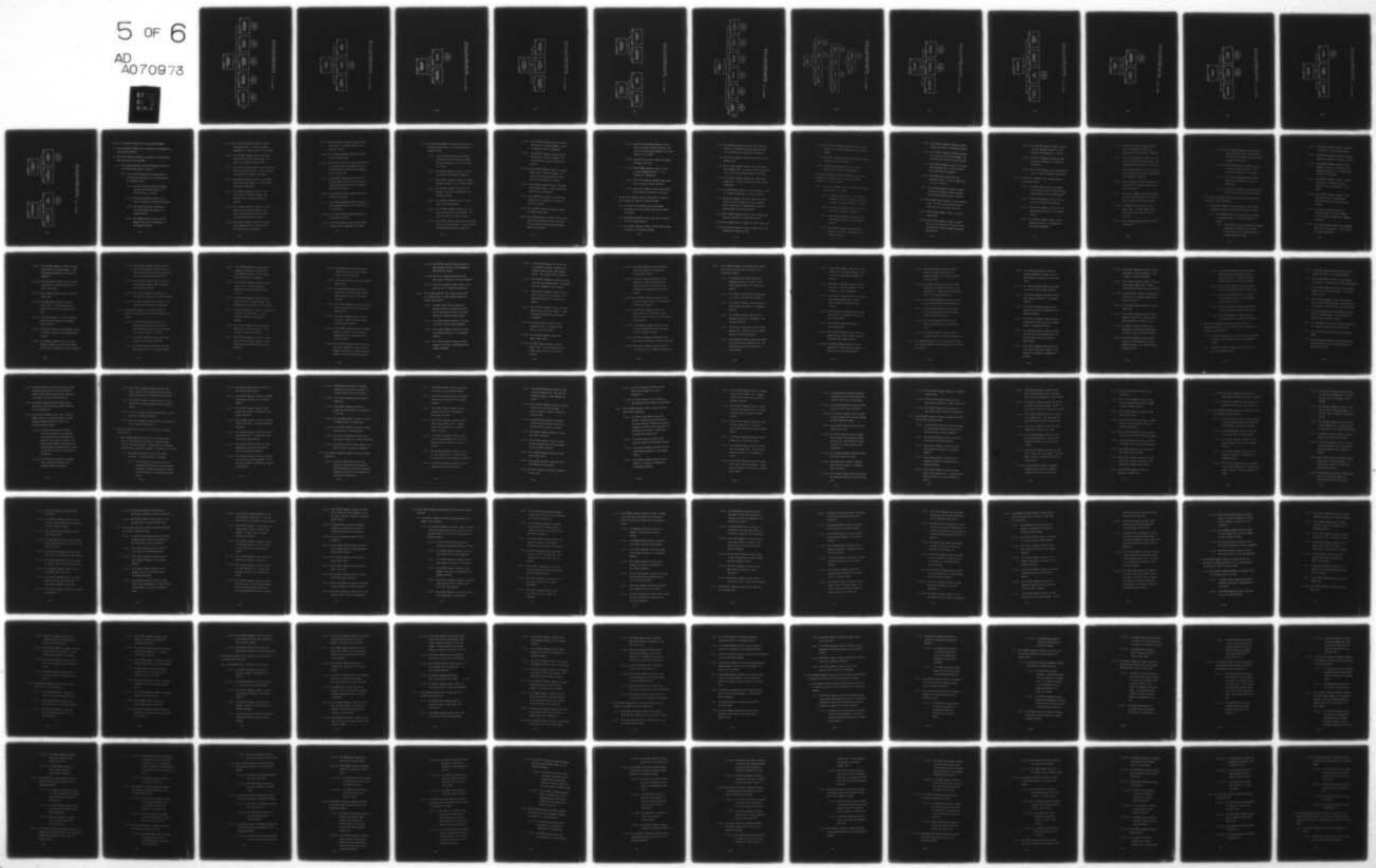


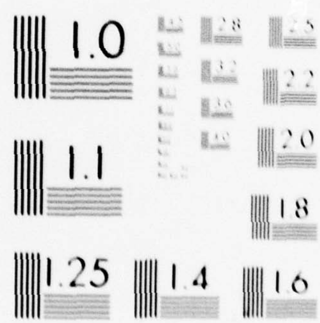
Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (55 of 68)

AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955

UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

5 OF 6
AD
A070973





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

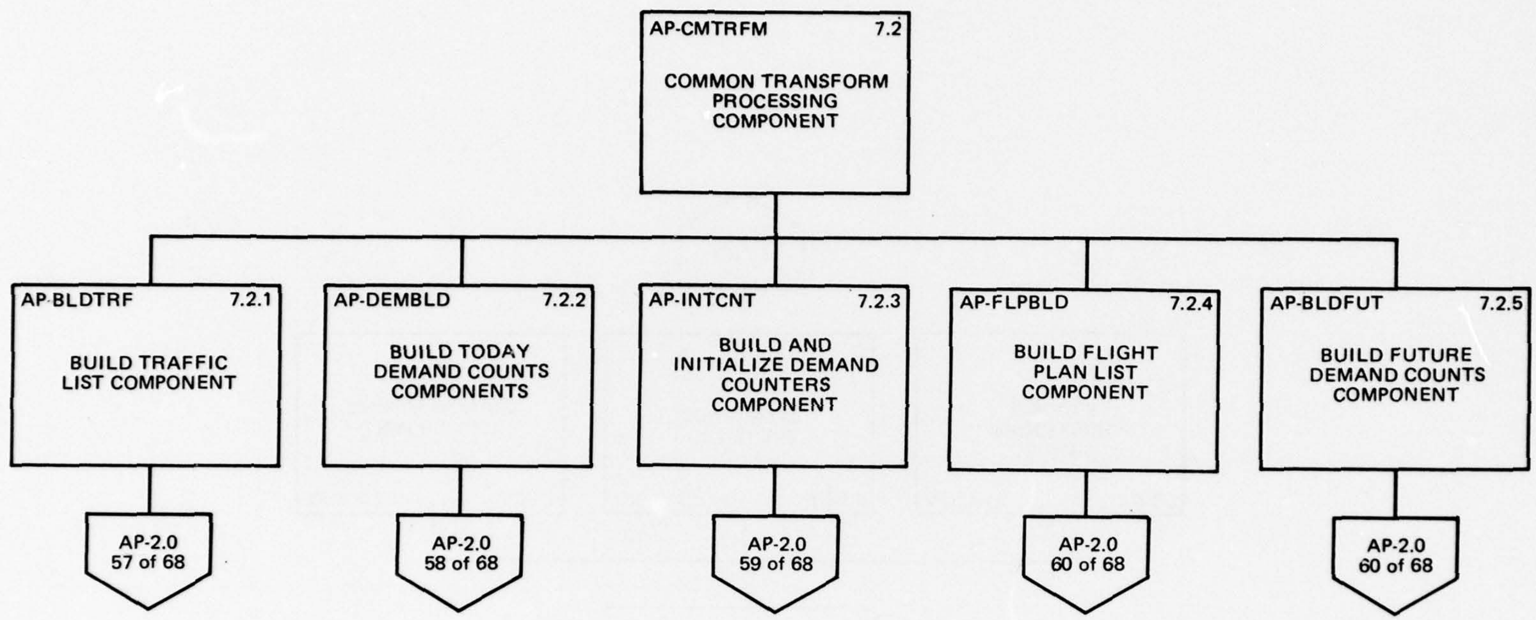


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (56 of 68)

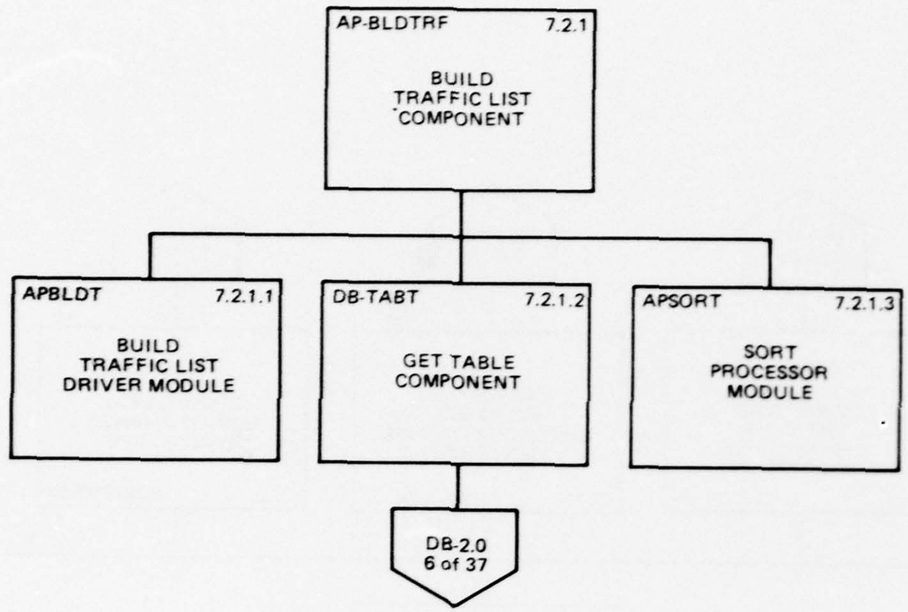


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (57 of 68)

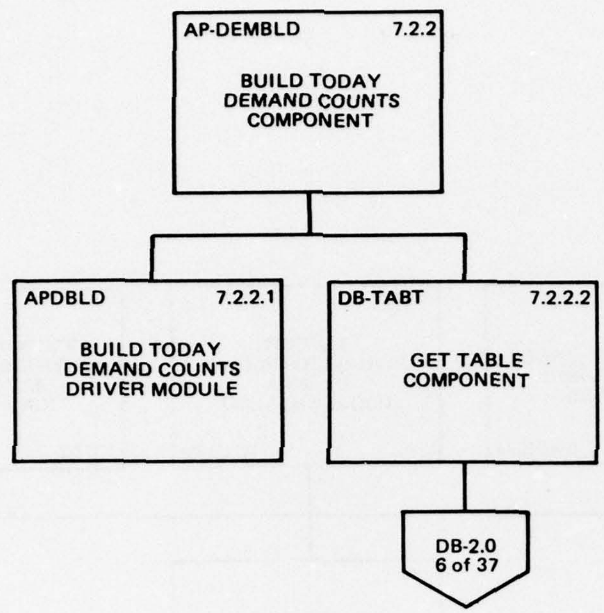


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (58 of 68)

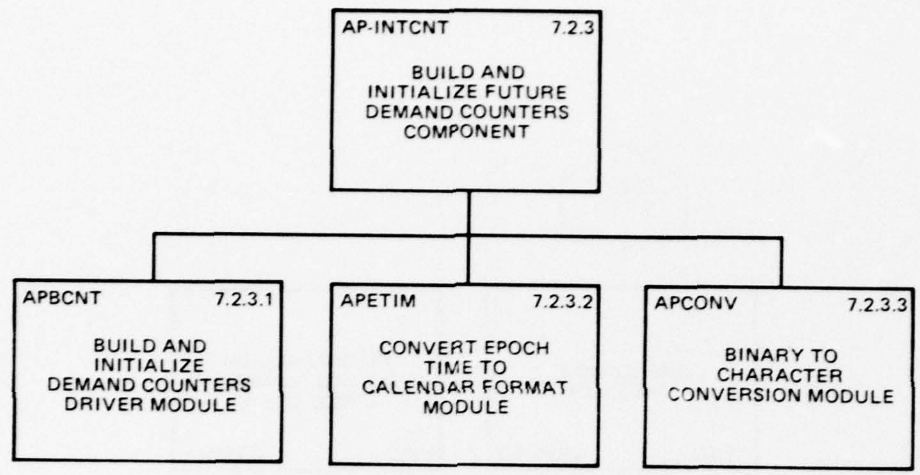


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (59 of 68)

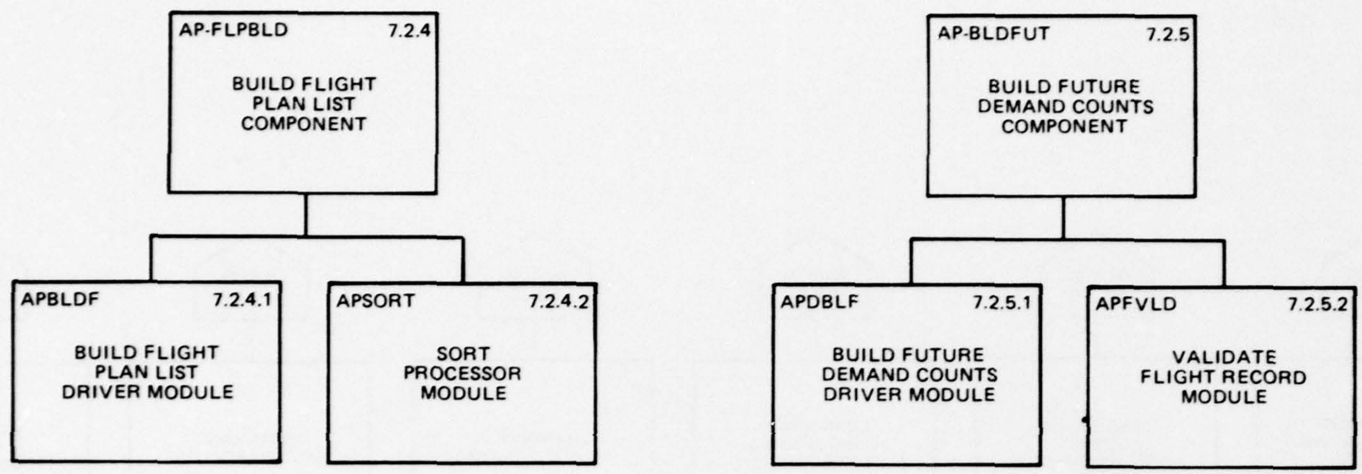


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (60 of 68)

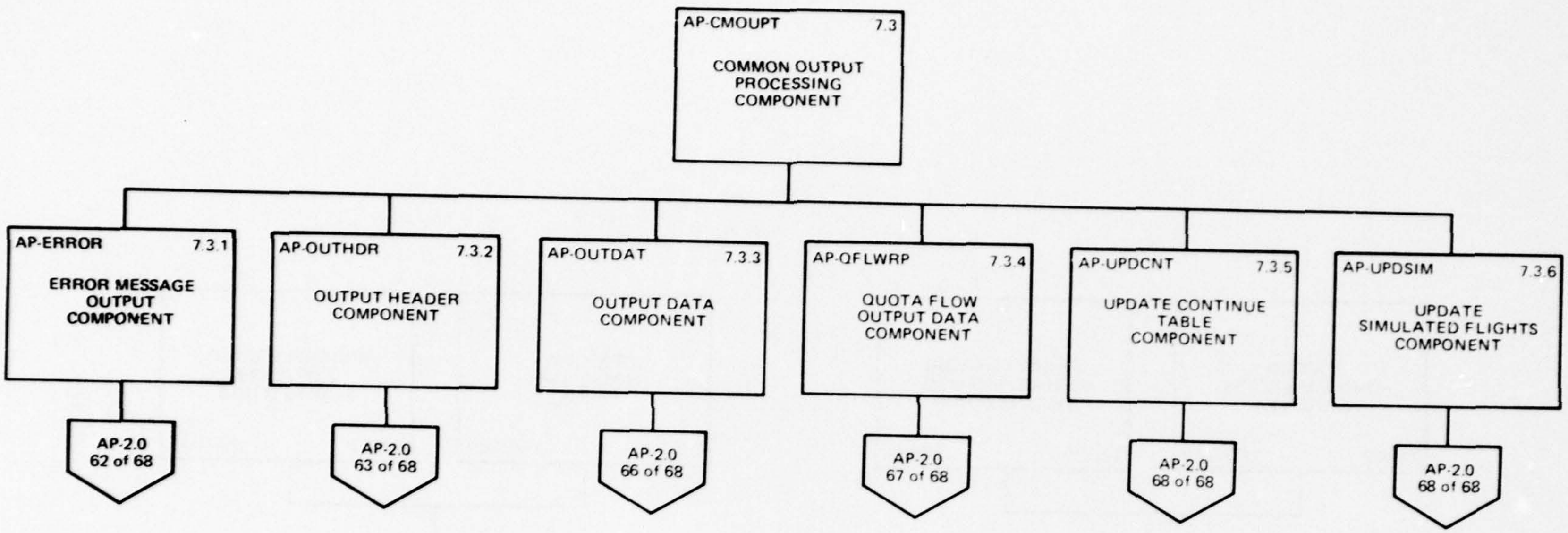


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (61 of 68)

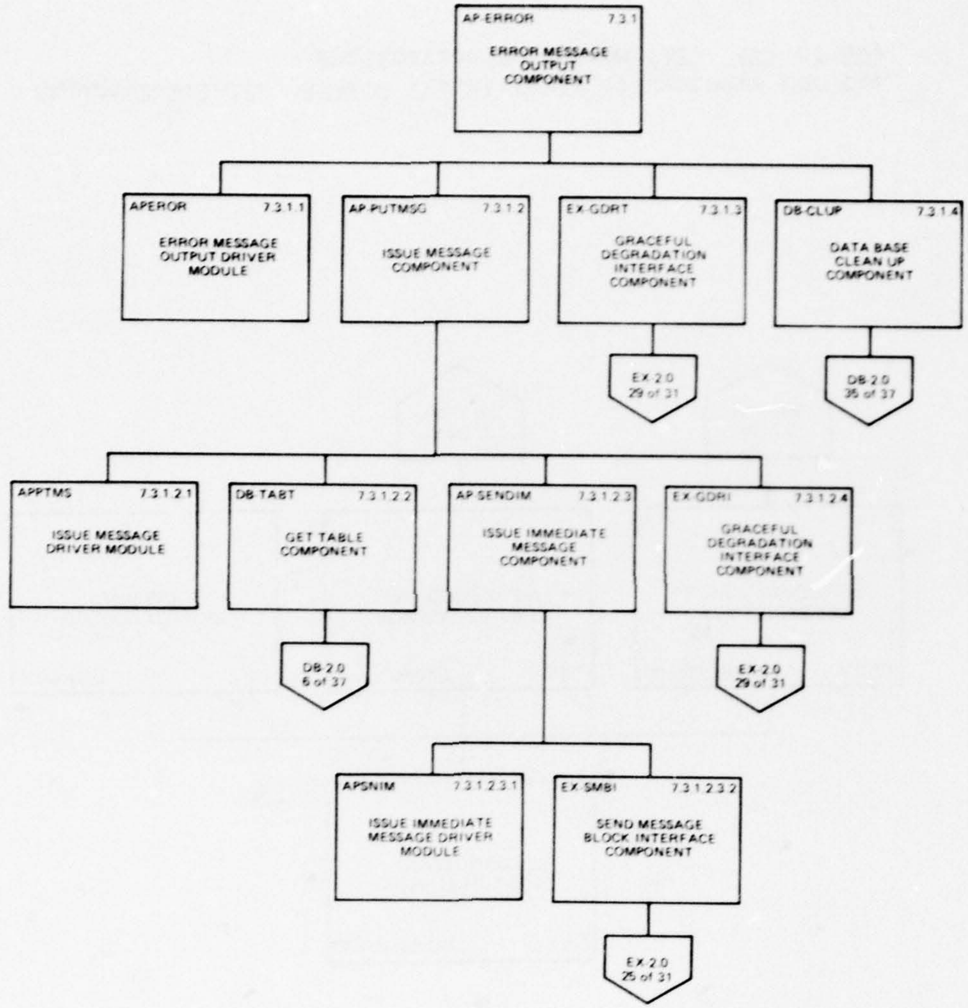


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (62 of 68)

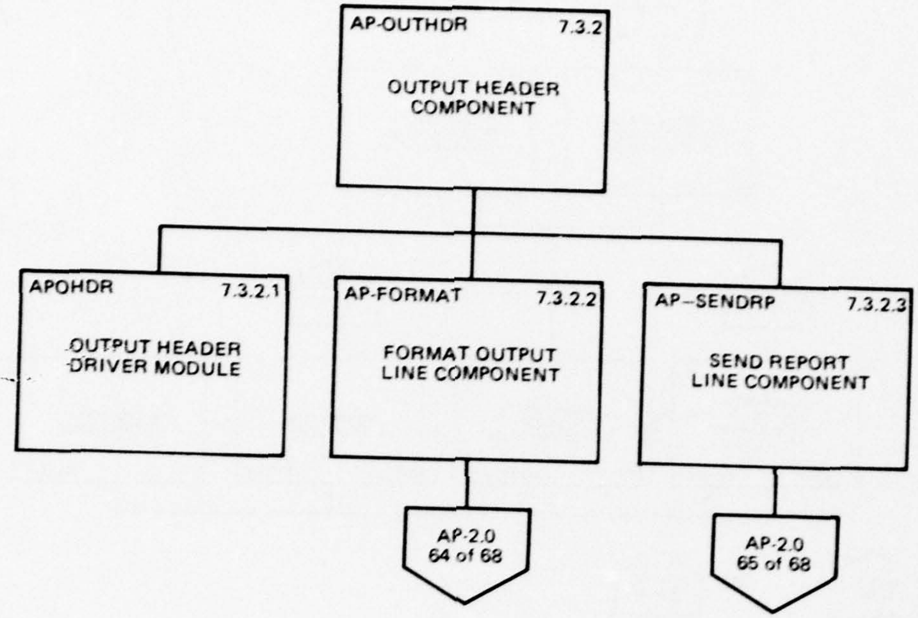


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (63 of 68)

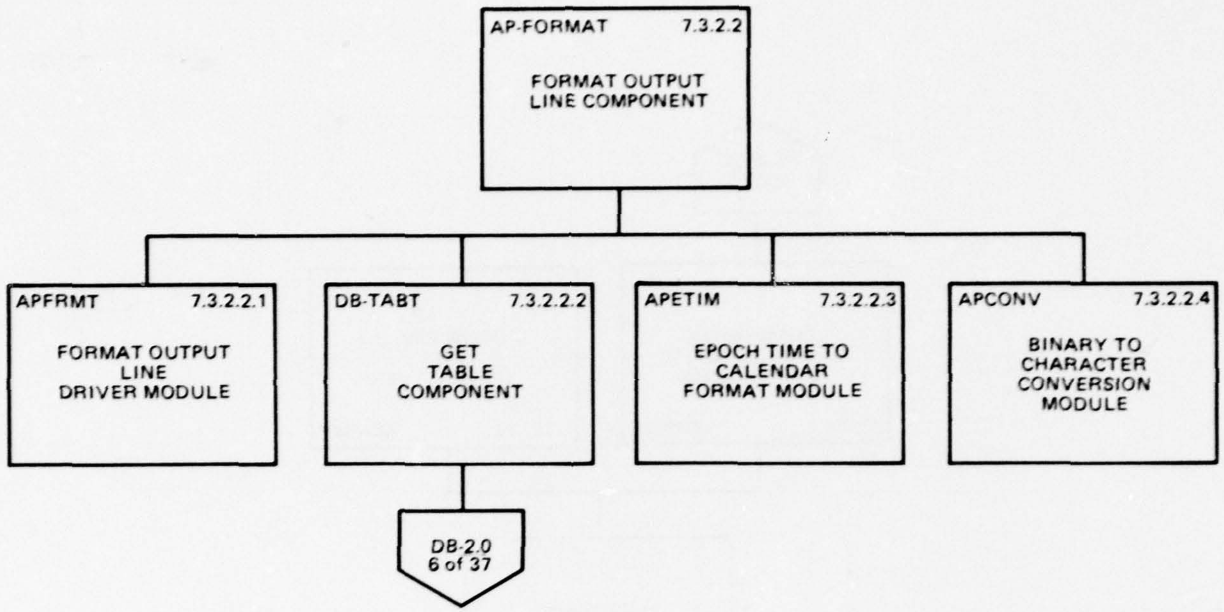


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (64 of 68)

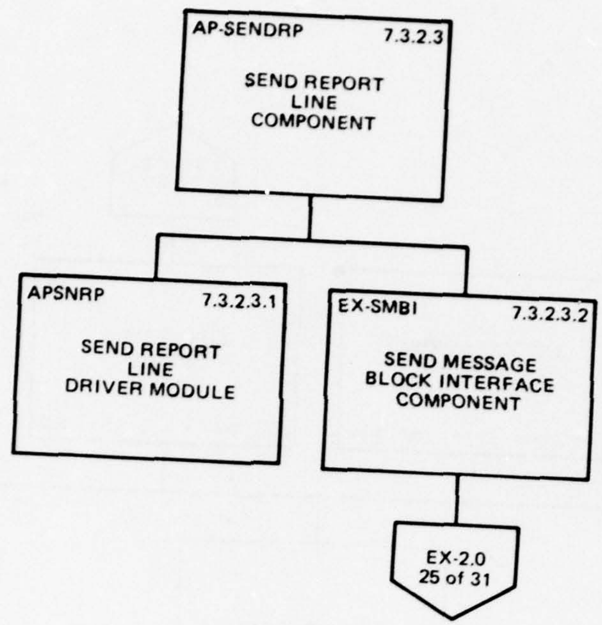


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (65 of 68)

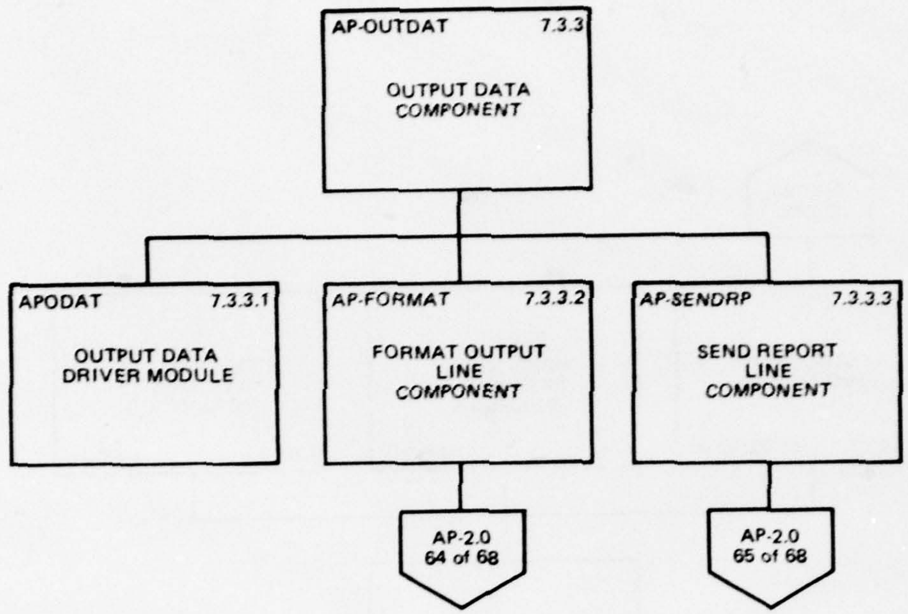


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (66 of 68)

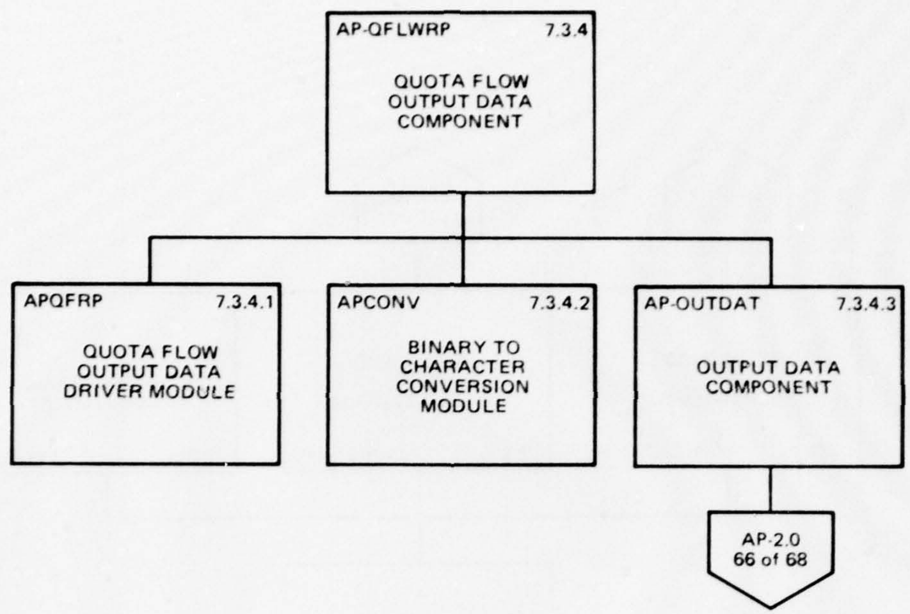


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (67 of 68)

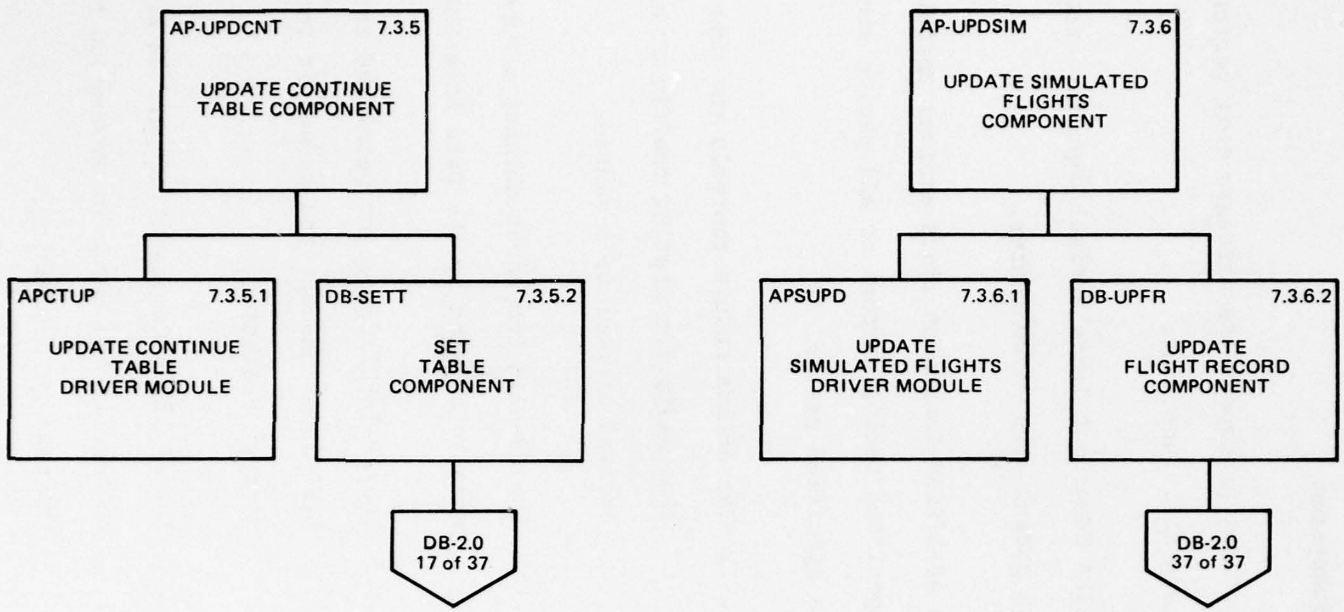


Figure 2.2.1-13. AP-2.0 Visual Table of Contents for the Application Subsystem (AP) (68 of 68)

2.2.1.4.11 Description Section for the Application Subsystem

2.0 The AP Subsystem processes the 26 messages that are passed from the Executive Subsystem.

3.0 The AP-LIST Component processes the messages that perform data base retrieval and list functions.

3.1 The AP-LISTRA Component lists arrival/departure traffic at a specified pacing airport or center.

3.1.1 The AP-LISTAR Component lists arrival traffic at a specified pacing airport or all pacing airports in a specified center.

3.1.1.1 The APLISA Module controls the message processing for listing traffic at an arrival airport or a center.

3.1.1.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

3.1.1.3 The AP-INPUT Component retrieves the LISA input message and parses it into message elements.

3.1.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

3.1.1.5 The AP-CTRPC2 Component converts a center/pacing place field. An entered center is converted to all pacing airports in the center.

3.1.1.6 The AP-STRSTP Component converts the start-stop time field of the input message. The inputs are converted to epoch seconds.

3.1.1.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is generated.

3.1.1.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types or Classes is generated.

3.1.1.9 The AP-ETECUT Component converts the ETE cutoff field of the input message. The converted time is output in minutes and seconds.

3.1.1.10 The AP-OUTFMT Component converts the output format field of the input message. The report title line and an array of report data line item positions are output.

3.1.1.11 The AP-OUTDEV Component converts the output device message field. A mask indicating the output devices is generated.

3.1.1.12 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

3.1.1.13 The AP-OUTHDR Component formats and outputs the report header lines.

3.1.1.14 The DB-GTFR Component retrieves flight records from the OAG and non-OAG flight data files for the specified affiliation and equipment types, and time span.

3.1.1.15 The AP-BLDTRF Component builds the list of arrival flights, sorted on arrival time, for the specified pacing airport or all pacing airports in the center.

3.1.1.16 The AP-OUTDAT Component formats and outputs the report data lines.

3.1.1.17 The AP-PUTMSG Component formats an accept or reject message and sends it to the Executive for transmittal.

3.1.1.18 The DB-CLUM Component performs normal Data Base clean-up functions.

3.1.1.19 The EX-RTDI Component accepts normal return control from the transaction load module.

3.1.2 The AP-LISTDP Component lists departure traffic at a specified pacing airport or all pacing airports in a specified center.

3.1.2.1 The APLISD Module controls the message processing for listing departure traffic at a pacing airport or all pacing airports in a center.

3.1.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

3.1.2.3 The AP-INPUT Component retrieves the input message and parses it into message elements.

3.1.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

3.1.2.5 The AP-CTRPC2 Component converts the place field of the input message.

3.1.2.6 The AP-STRSTP Component converts the start-stop time field of the input message. The times are converted to epoch seconds.

3.1.2.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is generated.

3.1.1.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types and Classes is generated.

3.1.2.9 The AP-ETECUT Component converts the ETE cutoff field of the input message. The converted time is output in minutes and seconds.

3.1.2.10 The AP-OUTFMT Component converts the output format field of the input message. The report title line and an array of report data line item positions are output.

3.1.2.11 The AP-OUTDEV Component converts the output device message field. A mask indicating the output devices is generated.

3.1.2.12 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

3.1.2.13 The AP-OUTHDR Component formats and outputs the report header lines.

3.1.2.14 The DB-GTFR Component retrieves flight records from the OAG and non-OAG flight data files for the specified affiliation and equipment types, and time span.

3.1.2.15 The AP-BLDTRF Component builds the list of departure flights, sorted on departure time, for the specified pacing airport or all pacing airports in the center.

3.1.2.16 The AP-OUTDAT Component formats and outputs the report data lines.

3.1.2.17 The AP-PUTMSG Component formats an accept or reject message and sends it to the Executive for transmittal.

3.1.2.18 The DB-CLUP Component performs normal termination Data Base clean-up functions.

3.1.2.19 The EX-RTDI Component accepts normal return control from the transaction load module.

3.2 The AP-LISTFL Component controls the process of creating a list of flight plan legs for a specified flight.

3.2.1 The APLIFP Driver Module controls the message processing for listing flight plan legs for a specified flight.

3.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

3.2.3 The AP-INPUT Component retrives the LIFP input message and parses it into message elements.

3.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing. For a critical error, processing is terminated.

3.2.5 The AP-ACID Component converts the aircraft ID input message field.

3.2.6 The AP-PACALL Component converts the pacing airport input message field. If the place code is defaulted, a list of all system pacing airports is generated.

3.2.7 The AP-OUTDEV Component converts the output device message field. A mask indicating the output devices is generated.

3.2.8 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

3.2.9 The DB-GTFR Component retrieves flight records from the OAG and Non-OAG flight files for a specified aircraft arriving at or departing from a specified pacing airport or all pacing airports.

3.2.10 The AP-FLPBLD Component builds a list of flight plan data for the report for a specified aircraft.

3.2.11 The AP-OUTHDR Component outputs the report header line.

3.2.12 The AP-OUTDAT Component outputs the report data lines containing the flight plan legs.

3.2.13 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

3.2.14 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

3.2.15 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

3.3 The AP-LISTAB Component lists either General Aviation (GA) estimates for a pacing airport or center, or landing capacities for a pacing airport.

3.3.1 The AP-LISTGA Component lists GA estimates for either a pacing airport or a center.

3.3.1.1 The APGAEL Module controls the process of generating a report of normal and today GA estimates for a pacing airport or a center.

3.3.1.2 The DB-WAIM Component obtains a work area for use by the Data Base Subsystem.

3.3.1.3 The AP-INPUT Component retrieves the GAEL input message and parses it into message elements.

3.3.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

3.3.1.5 The AP-CTRPC1 Component converts a center/pacing place field. No default is allowed.

3.3.1.6 The AP-STRSTP Component converts the start-stop time field of the input message. The input hours are converted to epoch seconds.

3.3.1.7 The AP-OUTDEV Component converts the output device message field. A mask indicating the output devices is generated.

3.3.1.8 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

3.3.1.9 The DB-TABT Component retrieves the normal and today GA estimates from the Data Base for the entered pacing airport or center.

3.3.1.10 The APHOUR Module produces an array of report hours. The hours range from the entered start hour up to the entered stop hour.

3.3.1.11 The AP-OUTHDR Component outputs the report header lines.

3.3.1.12 The AP-OUTDAT Component outputs the report data lines which contain the report hour and the normal and today GA estimates associated with the hour.

3.3.1.13 The AP-PUTMSG Component formats an accept or reject message and sends it to the Executive for transmittal to the user.

3.3.1.14 The DB-CLUP Component performs any Data Base clean-up required on a normal TLM termination.

3.3.1.15 The EX-RTDI Component accepts normal return control from the transaction load module.

3.3.2 The AP-LISTCP Component lists landing capacities for a pacing airport.

3.3.2.1 The APCAPL Module controls the process of generating a report of normal and today landing capacities for a pacing airport.

3.3.2.2 The DB-WAIM Component obtains a work area for use by the Data Base Subsystem.

3.3.2.3 The AP-INPUT Component retrieves the CAPL input message and parses it into message elements.

3.3.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

3.3.2.5 The AP-PACALL converts a pacing airport field. If defaulted, a list of all pacing airports in the System is generated.

3.3.2.6 The AP-STRSTP Component converts the start-stop time field of the input message. The input hours are converted to epoch seconds.

3.3.2.7 The AP-OUTDEV Component converts the output device message field. A mask indicating the output devices is generated.

3.3.2.8 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

3.3.2.9 The DB-TABT Component retrieves the normal and today landing capacities from the Data Base for the entered pacing airport(s).

3.3.2.10 The APHOUR Module produces an array of report hours. The hours range from the entered start hour up to the entered stop hour.

3.3.2.11 The AP-OUTHDR Component outputs the report header lines.

3.3.2.12 The AP-OUTDAT Component outputs the report data lines which contain the report hour and the normal and today landing capacities associated with the hour.

3.3.2.13 The AP-PUTMSG Component formats an accept or reject message and sends it to the Executive for transmittal to the user.

3.3.2.14 The DB-CLUP Component performs any Data Base clean-up required by a normal TLM termination.

3.3.2.15 The EX-RTDI Component accepts normal return control from the transaction load module.

4.0 The AP-COUNT Component performs data base retrieval and generates statistical count information.

4.1 The AP-REPDMD Component computes current or future arrival or departure statistics for a specified airport or center.

4.1.1 The AP-TARRDM Component lists hourly arrival counts for a pacing airport or all pacing airports in a center.

4.1.1.1 The APDEMA Driver Module controls the message processing to produce a list of hourly arrival counts for a pacing airport or all pacing airports in a center.

4.1.1.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.1.1.3 The AP-INPUT Component retrieves the input message and parses it into messa... elements.

4.1.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing. For a critical error, processing is terminated.

4.1.1.5 The AP-CTRPC2 Component converts a center/pacing field. An entered center is converted to all pacing airports in the center.

4.1.1.6 The AP-STRSTP Component converts a start-stop time field of an input message. The entered hours are converted to epoch seconds.

4.1.1.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is produced.

4.1.1.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types or Classes is generated.

4.1.1.9 The AP-OUTDEV Component converts the output device field of the input message. A mask indicating the specified output devices is generated.

4.1.1.10 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

4.1.1.11 The AP-OUTHDR Component outputs the report header lines.

4.1.1.12 The APHOUR Module produces an array of report hours. The hours range from the entered start hour up to the entered stop hour.

4.1.1.13 The DB-TABT Component is called to retrieve general aviation estimates for specified pacing airports.

4.1.1.14 The DB-GTFR Component retrieves flight records for flights arriving at the requested pacing airports.

4.1.1.15 The AP-DEMBLD Component uses the retrieved flights to produce a list of arrival demand counts for the specified start-stop time period.

4.1.1.16 The AP-OUTDAT Component outputs the report data lines consisting of hourly counts of arrival aircraft at the specified pacing airport or center (pacing airports only).

4.1.1.17 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.1.1.18 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.1.1.19 The EXRTDI Component returns control to the Executive after a normal TLM termination.

4.1.2 The AP-TDEPDM Component lists hourly departure counts for a pacing airport or all pacing airports in a center.

4.1.2.1 The PADEMD Driver Module controls the message processing to produce a list of hourly departure counts for a pacing airport or all pacing airports in a center.

4.1.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.1.2.3 The AP-INPUT Component retrieves the input message and parses it into message elements.

4.1.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing. For a critical error, processing is terminated.

4.1.2.5 The AP-CTRPC2 Component converts a center/pacing field. An entered center is converted to all pacing airports in the center.

4.1.2.6 The AP-STRSTP Component converts a start-stop time field of an input message. The entered hours are converted to epoch seconds.

4.1.2.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is produced.

4.1.2.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types or Classes is generated.

4.1.2.9 The AP-OUTDEV Component converts the output device field of the input message. A mask indicating the specified output devices is generated.

4.1.2.10 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

4.1.2.11 The AP-OUTHDR Component outputs the report header lines.

4.1.2.12 The APHOUR Module produces an array of report hours. The hours range from the entered start hour up to the entered stop hour.

4.1.2.13 The DB-TABT Component is called to retrieve general aviation estimates for specified pacing airports.

4.1.2.14 The DB-GTFR Component retrieves flight records for flights departing from the requested pacing airports.

4.1.2.15 The AP-DEMBLD Component uses the retrieved flights to produce a list of departure demand counts for the specified start-stop time period.

4.1.2.16 The AP-OUTDAT Component outputs the report data lines consisting of hourly counts of departure aircraft at the specified pacing airport or center (pacing airports only).

4.1.2.17 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.1.2.18 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.1.2.19 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

4.1.3 The AP-DDLYDM Component lists hourly arrival counts at a pacing airport for those flights delayed from another pacing airport.

4.1.3.1 The APDLDY Driver Module controls the message processing to produce the hourly list of demand arrival counts for flights delayed at a departure pacing airport.

4.1.3.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.1.3.3 The AP-INPUT Component retrieves the DDLY input message and parses it into message elements.

4.1.3.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

4.1.3.5 The AP-PACING Component is called twice to validate and convert the arrival and departure pacing airport input message fields. Both airports must be present.

4.1.3.6 The AP-STRSTP Component converts the start-stop time input message fields. The entered hours are converted to epoch seconds.

4.1.3.7 The AP-DELAYF Component converts the delay factor field of the input message. The departure delay factor is converted to seconds.

4.1.3.8 The AP-OUTDEV Component converts the output device field of the input message. A mask indicating the specified output devices is generated.

4.1.3.9 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

4.1.3.10 The AP-OUTHDR Component outputs the report header lines.

4.1.3.11 The APHOUR Module produces an array of report hours. The hours range from the entered start hour up to the entered stop hour.

4.1.3.12 The DB-TABT Component retrieves general aviation estimates for the specified arrival pacing airport.

4.1.3.13 The DB-GTFR Component retrieves flight records for flights departing from the specified departure pacing airport and arriving at the specified arrival pacing airport.

4.1.3.14 The AP-DEMBLD Component produces a list of arrival demand counts for the specified start-stop time period.

4.1.3.15 The AP-OUTDAT Component outputs the report data lines consisting of hourly counts of arrival aircraft at the specified pacing airport.

4.1.3.16 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.1.3.17 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.1.3.18 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

4.1.4 The AP-FARRDM Component lists hourly arrival counts for a future time period for any airport or all airports in a center.

4.1.4.1 The APDESA Driver Module controls the message processing to produce a list of hourly arrival counts for a future time period.

4.1.4.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.1.4.3 The AP-INPUT Component retrieves the DESA input message and parses it into message elements.

4.1.4.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

4.1.4.5 The AP-CTRAPT Component converts an input message field containing a center, pacing airport, or non-pacing airport.

4.1.4.6 The AP-SSDITM Component converts the start-stop, date-time input message field. The start and stop values are converted to epoch seconds.

4.1.4.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is produced.

4.1.4.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types or Classes is generated.

4.1.4.9 The AP-OUTDEV Component converts the output device input message field. A mask is created showing the specified output devices.

4.1.4.10 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

4.1.4.11 The AP-OUTHDR Component outputs the report header lines.

4.1.4.12 The AP-INTCNT Component builds a table of arrival counters for the report period and initializes the counters to zero.

4.1.4.13 The DB-TABT Component retrieves the GA estimates for a specified pacing airport or center, not for a non-pacing airport.

4.1.4.14 The DB-GTFR Component retrieves flight records for flights arriving at the specified airport or all flights arriving within the center.

4.1.4.15 The AP-BLDFVT Component counts the flights arriving at the specified place code on an hourly basis. A flight could arrive several days out of the specified period.

4.1.4.16 The AP-OUTDAT Component outputs the report data lines containing arrival demand counts.

4.1.4.17 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.1.4.18 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.1.4.19 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

4.1.5 The AP-FDEPDM Component lists hourly departure counts for a future time period for any airport or all airports in a center.

4.1.5.1 The APDESD Driver Module controls the message processing to produce a list of hourly departure counts for a future time period.

4.1.5.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.1.5.3 The AP-INPUT Component retrieves the DESA input message and parses it into message elements.

4.1.5.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

4.1.5.5 The AP-CTRAPT Component converts an input message field containing a center, pacing airport, or non-pacing airport.

4.1.5.6 The AP-SSDTTM Component converts the start-stop, date-time input message field. The start and stop values are converted to epoch seconds.

4.1.5.7 The AP-AFFIL Component converts the affiliation field of the input message. A list of Airline or Operational Categories is produced.

4.1.5.8 The AP-EQUIP Component converts the equipment field of the input message. A list of Types or Classes is generated.

4.1.5.9 The AP-OUTDEV Component converts the output device input message field. A mask is created showing the specified output devices.

4.1.5.10 The APETIM Module converts the epoch transaction time to a calendar format which is used in the report.

4.1.5.11 The AP-OUTHDR Component outputs the report header lines.

4.1.5.12 The AP-INTCNT Component builds a table of departure counters for the report period and initializes the counters to zero.

4.1.5.13 The DB-TABT Component retrieves the GA estimates for a specified pacing airport or center; not for a non-pacing airport.

4.1.5.14 The DB-GTFR Component retrieves flight records for flights departing from the specified airport or all flights arriving within the center.

4.1.4.15 The AP-BLDFUT Component counts the flights departing from the specified place code on an hourly basis. A flight could depart several days out of the specified period.

4.1.4.16 The AP-OUTDAT Component outputs the report data lines containing departure demand counts.

4.1.5.17 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.1.5.18 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.1.5.19 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

4.2 The AP-COMFIX Component lists hourly arrival counts at a specified arrival fix or at all arrival fixes associated with a specified pacing airport.

4.2.1 The APFIXL Driver Module controls the message processing to produce a list of arrival counts at the requested fix(es).

4.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

4.2.3 The AP-INPUT Component retrieves the FIXL input message and parses it into message elements.

4.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

4.2.5 The AP-FIXPAC Component converts the fix or pacing airport field of the input message. A vector of fixes is output.

4.2.6 The AP-STRSTP Component converts the start-stop time field of the input message. The entered hours are converted to epoch seconds for the current day.

4.2.7 The AP-OUTDEV Component converts the output device field of the input message. A mask showing the specified output devices is output.

4.2.8 The APETIM Module converts the transaction epoch time to a calendar format to be used in the report.

4.2.9 The AP-OUTHDR Component outputs the report header lines.

4.2.10 The APHOUR Module computes an array of report hours. The hours range from the entered start hour up to the entered stop hour.

4.2.11 The DB-TABT Component retrieves the arrival pacing airport associated with a fix and all departure airports which release flights passing through the fix, enroute to the arrival pacing airport.

4.2.12 The DB-GTFR Component retrieves flight records arriving at the arrival pacing airport associated with the specified fix.

4.2.13 The AP-FIXBLD Component builds a list of arrival demand counts at the specified fix(es) for those retrieved flights departing from one of the retrieved departure airports.

4.2.13.1 The APFXBD Module creates a table of fix arrival counters for the specified report period and then initializes the counters to zero (first call only). For each retrieved flight record meeting the arrival/departure qualifications, a counter for the hour of arrival at the fix(es) is incremented.

4.2.13.2 The APCONV Module converts the computed report hours from internal binary to character format for printing.

4.2.14 The AP-OUTDAT Component outputs the report data lines. Hourly counts of arrivals at the specified fix(es) are output for the specified report period.

4.2.15 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

4.2.16 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

4.2.17 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.0 The AP-SIM Component performs data base retrieval and simulates flight traffic.

5.1 The AP-PERSIM Component performs an arrival demand, quota flow, or fuel advisory simulation for a specified pacing airport. Data base updating of flight records and simulation continue data is performed if flights are assigned flow controls.

5.1.1 The AP-ARDSIM performs an arrival delay (ARRD) simulation. No data base update is performed.

5.1.1.1 The APARRD Module controls the process of validating and converting the input message, invoking the Simulation Subsystem to perform the ARRD simulation, and generating the ARRD report.

5.1.1.2 The DB-WAIM Component obtains a work area for the Data Base Subsystem.

5.1.1.3 The AP-INPUT Component retrieves the APARRD input message and parses it into message elements.

5.1.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.1.5 The AP-PACING Component converts the pacing airport place field. A pacing airport must be entered.

5.1.1.6 The AP-STRSTP Component converts the start-stop message field. The input hours are converted into epoch seconds.

5.1.1.7 The AP-STACK Component converts the input message stack time and size fields for use by the Simulation Subsystem.

5.1.1.8 The AP-OUTDEV Component converts the output device field of the input message. An output mask is created to indicate the selected output devices.

5.1.1.9 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.1.10 The SI-ARRD Component performs the ARRD simulation.

5.1.1.11 The AP-PUTMSG Component retrieves a message from the Data Base for transmittal to the user.

5.1.1.12 The AP-OUTHDR Component formats and outputs the header lines of the ARRD report.

5.1.1.13 The AP-OUTDAT Component formats and outputs the data lines of the ARRD report.

5.1.1.14 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.1.15 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.1.2 The AP-QFLWSM Component performs a quota flow (QFLW) simulation.

5.1.2.1 The APQFLW Driver Module controls the process of validating and converting the QFLW input message, invoking the Simulation Subsystem to perform a QFLW simulation, and generating the QFLW report.

5.1.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

5.1.2.3 The AP-INPUT Component retrieves the QFLW input message and parses it into message elements.

5.1.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.2.5 The AP-PACING Component converts a pacing airport input message field. A pacing airport must be entered or a rejection message is issued.

5.1.2.6 The AP-STRSTP Component converts a start-stop time input message field. The start and stop hours are converted to epoch seconds.

5.1.2.7 The AP-ZONEID Component converts the zone ID input message field. The zone ID must be associated with the entered pacing airport.

5.1.2.8 The AP-STACK Component converts the input message stack time and size fields.

5.1.2.9 The AP-HOLDTM Component converts the hold time input message field. The hold time entered in minutes is also converted to seconds.

5.1.2.10 The AP-OUTDEV Component converts the output device field of the input message. A mask is created to indicate the selected output devices.

5.1.2.11 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.2.12 The SI-QFLOW Component performs the quota flow (QFLW) simulation.

5.1.2.13 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

5.1.2.14 The AP-OUTHDR Component outputs the QFLW report header lines.

5.1.2.15 The AP-QFLWRP Component outputs the quota flow (QFLW) report data lines.

5.1.2.16 The UPDCNT Component updates the Simulation Continue Table.

5.1.2.17 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.2.18 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.1.3 The AP-QFLZSM Component performs a quota flow with origin (QFLZ) simulation.

5.1.3.1 The APQFLZ Driver Module controls the process of validating and converting the QFLZ input message, invoking the Simulation Subsystem to perform a QFLZ simulation, and generating a QFLZ report (a QFLW report with additional origin data).

5.1.3.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

5.1.3.3 The AP-INPUT Component retrieves the QFLZ input message and parses it into message elements.

5.1.3.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.3.5 The AP-PACING Component converts a pacing airport input message field. A pacing airport must be entered or a rejection message is issued.

5.1.3.6 The AP-STRSTP Component converts a start-stop time input message field. The start and stop hours are converted to epoch seconds.

5.1.3.7 The AP-ZONEID Component converts the zone ID input message field. The zone ID must be associated with the entered pacing airport.

5.1.3.8 The AP-STACK Component converts the input message stack time and size fields.

5.1.3.9 The AP-HOLDTM Component converts the hold time input message field. The hold time entered in minutes is also converted to seconds.

5.1.3.10 The AP-OUTDEV Component converts the output device field of the input message. An output mask is created to indicate the selected output devices.

5.1.3.11 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.3.12 The SI-QFLOW Component performs the quota flow with origin (QFLZ) simulation.

5.1.3.13 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

5.1.3.14 The AP-OUTHDR Component outputs the QFLZ report header lines.

5.1.3.15 The DB-TABT Component returns an origin center external code, given the center internal code. Module DBTABT is called for each of the origin centers in the origin report.

5.1.3.16 The AP-QFLWRP Component outputs the quota flow (QFLW) report data lines.

5.1.3.17 The APCONV Module converts the origin center departure counts to a character format for output.

5.1.3.18 The AP-OUTDAT Component outputs the report data lines associated with the QFLZ origin data.

5.1.3.19 The AP-UPDCNT Component updates the Simulation Continue Table.

5.1.3.20 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.3.21 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.1.4 The AP-FADFSM Component performs the message processing for the FADF simulation message.

5.1.4.1 The APFADF Driver Module controls the process of input message validation and conversion, FADF simulation, and report generation.

5.1.4.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

5.1.4.3 The AP-INPUT Component retrieves the FADF input message and parses it into message elements.

5.1.4.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.4.5 The AP-PACING Component converts a pacing airport input message field. A pacing airport must be entered or a rejection message is issued.

5.1.4.6 The AP-STRSTP Component converts a start-stop time input message field. The start and stop hours are converted to epoch seconds.

5.1.4.7 The AP-ETECUT Component converts the ETE cut-off field of the input message. The ETE input value in minutes is converted to seconds.

5.1.4.8 The AP-ZONEID Component converts the zone ID input message field. The zone ID must be associated with the entered pacing airport.

5.1.4.9 The AP-STACK Component converts the input message stack time and size fields.

5.1.4.10 The AP-HOLDTM Component converts the hold time input message field. The hold time entered in minutes is also converted to seconds.

5.1.4.11 The AP-OUTDEV Component converts the output device field of the input message. An output mask is created to indicate the selected output devices.

5.1.4.12 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.4.13 The SI-FAD Component performs the FADF simulation.

5.1.4.14 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

5.1.4.15 The AP-OUTHDR Component outputs the FADF report header lines.

5.1.4.16 The APHOUR Module builds an array of hourly values from the entered start hour up to the entered stop hour.

5.1.4.17 The AP-OUTDAT Component outputs the report data links for the various FADF type reports.

5.1.4.18 The DB-TABT Component retrieves the normal GA estimates for the entered pacing airport.

5.1.4.19 The AP-QFLWRP Component outputs the quota flow (QFLW) report data lines.

5.1.4.20 The AP-UPDSIM Component updates the flight records in the OAG and non-OAG files for those flights assigned flow controls.

5.1.4.21 The AP-UPDCNT Component updates the Simulation Continue Table.

5.1.4.22 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.4.23 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.1.5 The AP-FADPSM Component performs the message processing for the FADP simulation message.

5.1.5.1 The APFADP Driver Module controls the process of input message validation and conversion, FADP simulation, and report generation.

5.1.5.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

5.1.5.3 The AP-INPUT Component retrieves the FADP input message and parses it into message elements.

5.1.5.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.5.5 The AP-PACING Component converts a pacing airport input message field. A pacing airport must be entered or a rejection message is issued.

5.1.5.6 The AP-STRSTP Component converts a start-stop time input message field. The start and stop hours are converted to epoch seconds.

5.1.5.7 The AP-ETECUT Component converts the ETE cut-off field of the input message. The ETE input value in minutes is converted to seconds.

5.1.5.8 The AP-ZONEID Component converts the zone ID input message field. The zone ID must be associated with the entered pacing airport.

5.1.5.9 The AP-STACK Component converts the input message stack time and size fields.

5.1.5.10 The AP-HOLDTM Component converts the hold time input message field. The hold time entered in minutes is also converted to seconds.

5.1.5.11 The AP-OUTDEV Component converts the output device field of the input message. An output mask is created to indicate the selected output devices.

5.1.5.12 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.5.13 The SI-FAD Component performs the FADP simulation.

5.1.5.14 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

5.1.5.15 The AP-OUTHDR Component outputs the FADP report header lines.

5.1.5.16 The APHOUR Module builds an array of hourly values from the entered start hour up to the entered stop hour.

5.1.5.17 The AP-OUTDAT Component outputs the report data lines for the various FADP type reports.

5.1.5.18 The DB-TABT Component retrieves the normal GA estimates for the entered pacing airport.

5.1.5.19 The AP-QFLWRP Component outputs the quota flow (QFLW) report data lines.

5.1.5.20 The AP-UPDSIM Component updates the flight records in the OAG and non-OAG files for those flights assigned flow controls.

5.1.5.21 The AP-UPDCNT Component updates the Simulation Continue Table.

5.1.5.22 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.5.23 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

5.1.6 The AP-FADTSM Component performs the message processing for the FADT simulation message.

5.1.6.1 The APFADT Driver Module controls the process of input message validation and conversion, FADT simulation, and report generation.

5.1.6.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

5.1.6.3 The AP-INPUT Component retrieves the FADT input message and parses it into message elements.

5.1.6.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

5.1.6.5 The AP-PACING Component converts a pacing airport input message field. A pacing airport must be entered or a rejection message is issued.

5.1.6.6 The AP-STRSTP Component converts a start-stop time input message field. The start and stop hours are converted to epoch seconds.

5.1.6.7 The AP-TESTCP Component converts the test capacity update field of the FADT input message. An hourly array of entered capacities is created.

5.1.6.8 The AP-ZONEID Component converts the zone ID input message field. The zone ID must be associated with the entered pacing airport.

5.1.6.9 The AP-STACK Component converts the input message stack time and size fields.

5.1.6.10 The AP-HOLDTM Component converts the hold time input message field. The hold time entered in minutes is also converted to seconds.

5.1.6.11 The AP-DEPDLY Component converts the departure-delay field of the FADT input message. A table of entered pacing airports and their assigned departure delays is created.

5.1.6.12 The AP-OUTDEV Component converts the output device field of the input message. An output mask is created to indicate the selected output devices.

5.1.6.13 The APETIM Module converts the message transaction time from epoch seconds to a calendar format for use in the report.

5.1.6.14 The SI-FAD Component performs the FADT simulation.

5.1.6.15 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

5.1.6.16 The AP-OUTHDR Component outputs the FADT report header lines.

5.1.6.17 The AP-OUTDAT Component outputs the FADT report data lines.

5.1.6.18 The AP-QFLWRP Component outputs the quota flow (QFLW) report data lines.

5.1.6.19 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

5.1.6.20 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.0 The AP-UPDATE Component performs data base retrieval and update functions.

6.1 The AP-ALTFLT Component processes the messages which alter flight in the data base.

6.1.1 The AP-ACTFLT Component activates a flight or flights of the OAG flight record file for the period of the specified date to the end of the flight record file effective period.

6.1.1.1 The APACTV Driver Module controls the process of activating specified OAG flights.

6.1.1.2 The DB-WAIM Component obtains a work area to be used by the DAta Base Subsystem.

6.1.1.3 The AP-INPUT Component retrieves the ACTV input message and parses it into message

6.1.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.1.1.5 The AP-ACIDSG Component converts the Aircraft Identification/Airline Designator Field of the ACTV input message.

6.1.1.6 The AP-DATE Component converts the entered input message date to epoch seconds.

6.1.1.7 The DB-GTFR Component retrieves flight records from the OAG file for the specified ACID or Airline Designator.

6.1.1.8 The APCMSK Module changes the activity date mask for each filtered flight to indicate the flight is activated for the specified time period.

6.1.1.9 The APSORT Module sorts the flights to be updated on relative record number to efficiently perform the OAG file updating.

6.1.1.10 The DB-UPFR Component updates one flight record at a time in the Data Base. The activity mask of each flight record is replaced.

6.1.1.11 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.1.1.12 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.1.1.13 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.1.2 The AP-INHFLT Component inhibits a flight or flights of the OAG flight record file from the specified date to the end of the flight record file effective period.

6.1.2.1 The APINHB Driver Module controls the process of inhibiting specified OAG flights.

6.1.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.1.2.3 The AP-INPUT Component retrieves the INHB input message and parses it into message elements.

6.1.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.1.2.5 The AP-ACIDSG Component converts the Aircraft Identification/Airline Designator field of the ACTV input message.

6.1.2.6 The AP-DATE Component converts the entered input message date to epoch seconds.

6.1.2.7 The DB-GTFR Component retrieves flight records from the OAG file for the specified ACID or Airline Designator.

6.1.2.8 The APCMSK Module changes the activity date mask for each filtered flight to indicate the flight is inhibited for the specified time period.

6.1.2.9 The APSORT Module sorts the flights to be updated on relative record number to efficiently perform the OAG file updating.

6.1.2.10 The DB-UPFR Component is called once for each flight record to be updated. The activity mask of each specified flight record is replaced.

6.1.2.11 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.1.2.12 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.1.2.13 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.1.3 The AP-CXSFLT Component cancels an OAG flight leg for the current day.

6.1.3.1 The APCXSD Driver Module controls the process of cancelling an OAG flight leg for the current day.

6.1.3.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.1.3.3 The AP-INPUT Component retrieves the CXSD input message and parses it into message elements.

6.1.3.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.1.3.5 The AP-ACID Component converts the Aircraft Identification field of the CXSD input message.

6.1.3.6 The AP-PACNPC Component converts the departure airport (pacing or non-pacing) field of the input message.

6.1.3.7 The AP-HRMIN Component converts the hour-minute field of the input message. The hour and minute values are converted to epoch seconds of the current day.

6.1.3.8 The DB-GTFR Component retrieves flight records from the OAG flight record file for the specified flight (ACID).

6.1.3.9 The APFVLD Module determines if a flight is valid for a specified time period. It is called to determine a unique flight leg, given the specified planned gate time of departure.

6.1.3.10 The APCMSK Module changes the departure date mask to indicate the flight was cancelled for the current day.

6.1.3.11 The DB-UPFR Component updates the flight record to indicate the cancellation for the current day. The departure date mask is replaced.

6.1.3.12 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.1.3.13 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.1.3.14 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.1.4 The AP-RMVFLT Component cancels a flight from the OAG for the current day or deletes a flight from the non-OAG file.

6.1.4.1 The APRMER Driver Module controls the process of cancelling an OAG flight or deleting a non-OAG flight.

6.1.4.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.1.4.3 The AP-INPUT Component retrieves the RS input message and parses it into message elements.

6.1.4.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.1.4.5 The AP-ACID Component converts the Aircraft Identification field of the RS input message.

6.1.4.6 The AP-PACNPC Component converts the departure airport (pacing or non-pacing) field of the input message.

6.1.4.7 The AP-HRMIN Component converts the hour-minute field of the input message. The hour

and minute values are converted to epoch seconds of the current day.

6.1.4.8 The DB-GTFR Component retrieves flight records from the flight record files for the specified flight (ACID).

6.1.4.9 The APFVLD Module determines if a flight is valid for a specified time period. It is called to determine a unique flight leg, given the specified planned gate time of departure.

6.1.4.10 The DB-TABT Component retrieves the pacing airport indicator for the arrival airport of the filtered flight record. The flight to be cancelled or deleted must arrive at a pacing airport.

6.1.4.11 The APCMSK Module changes the departure date mask of the flight record. For an OAG flight, the current date bit position is turned off to indicate cancellation for today only. For non-OAG flights, all bit positions are turned off.

6.1.4.12 The DB-UPFR Component updates the flight record to indicate cancellation for today of an OAG flight or deletion for a non-OAG flight.

6.1.4.13 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.1.4.14 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.1.4.15 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.2 The AP-UPDTAB Component updates a general aviation (GA) estimates or landing capacities table for a pacing airport (or optionally a center for the GA table).

6.2.1 The AP-UPDGAT Component updates a GA estimates table for a pacing airport or a center.

6.2.1.1 The APGAES Module controls the process of updating a GA table for a pacing airport or a center.

6.2.1.2 The DB-WAIM Component obtains a work area for the Data Base Subsystem.

6.2.1.3 The AP-INPUT Component retrieves the input message and parses it into message elements.

6.2.1.4 The AP-ERROR Component outputs an error message for any error encountered in the message processing.

6.2.1.5 The AP-CTRPC1 Component converts a center/pacing place field. No default is allowed.

6.2.1.6 The AP-GAEUPD Component converts the GA update field of the input message. An hourly array of update values is produced.

6.2.1.7 The APETIM Module converts the epoch transaction time into a calendar date format for the report.

6.2.1.8 The DB-TABT Component retrieves the GA estimates for the pacing airport or center from the Data Base.

6.2.1.9 The AP-OUTHDR Component outputs the report header lines.

6.2.1.10 The AP-OUTDAT Component outputs the report data lines.

6.2.1.11 The DB-SETT Component updates the GA estimates table for the pacing airport or center in the Data Base.

6.2.1.12 The AP-PUTMSG Component retrieves a message from the Data Base and sends it to the Executive for transmittal to the user.

6.2.1.13 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.2.1.14 The EX-RTDI Component accepts normal return control from the transaction load module.

6.2.2 The AP-UPDCPT Component updates landing capacity table values for a pacing airport.

6.2.2.1 The APCAPS Driver Module controls the process of updating landing capacities.

6.2.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.2.2.3 The AP-INPUT Component retrieves the CAPS input message and parses it into message elements.

6.2.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.2.2.5 The AP-PACING Component converts the pacing airport input message field. A pacing airport must be entered.

6.2.2.6 The AP-CAPUPD Component converts the landing capacity update field of the input message.

6.2.2.7 The APETIM Module converts the message transaction time in epoch seconds to a calendar format for the report.

6.2.2.8 The DB-TABT Component retrieves the landing capacities (24) for the specified pacing airport.

6.2.2.9 The AP-OUTHDR Component outputs the landing capacity report header lines.

6.2.2.10 The AP-OUTDAT Component outputs the landing capacity report data lines.

6.2.2.11 The DB-SETT Component updates the landing capacities in the Data Base for the specified pacing airport.

6.2.2.12 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.2.2.13 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.3 The AP-ADDFLP Component adds a flight plan to either the OAG or non-OAG flight record file.

6.3.1 The AP-ADDFPO adds a flight plan to the OAG file.

6.3.1.1 The APFPSD Driver Module controls the process of adding a flight plan to the OAG file.

6.3.1.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.3.1.3 The AP-INPUT Component retrieves the FPSD input message and parses it into message elements.

6.3.1.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.3.1.5 The AP-ACID Component converts the Aircraft Identification field of the FPSD input message.

6.3.1.6 The AP-PACNPC Component converts the pacing/non-pacing airport field of the input message. The airport must be entered.

6.3.1.7 The AP-HRMIN Component converts the hour-minute field (PGTD) of the input message. The hour and minute values are converted to epoch seconds.

6.3.1.8 The AP-ETEVAL Component converts the estimated time enroute field of the input message.

6.3.1.9 The AP-TYPE Component converts the equipment type field of the FPSD input message. The associated aircraft class is returned.

6.3.1.10 The APDOWP Module converts the days of operation field of the FPSD input message. An array showing weekly days of operation is generated.

6.3.1.11 The AP-SSDATE Component converts the start-stop, date-time fields of the FPSD input message. The start and stop dates are converted to epoch seconds.

6.3.1.12 The APDOWM Module creates a departure date mask from the day of operation array created by Module APDOWP.

6.3.1.13 The DB-GTFR Component retrieves any flight records from the OAG corresponding to the flight information specified by the input message. A returned flight indicates the flight plan should not be added to the file.

6.3.1.14 The DB-UPFR Component adds the specified flight plan to the OAG flight record file.

6.3.1.15 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.3.1.16 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.3.1.17 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.3.2 The AP-ADDFPN Component adds a flight plan to the non-OAG flight record file.

6.3.2.1 The APFPER Driver Module controls the process of adding a flight plan to the non-OAG file.

6.3.2.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.3.2.3 The AP-INPUT Component retrieves the FP input message and parses it into message elements.

6.3.2.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.3.2.5 The AP-ACID Component converts the Aircraft Identification field of the FP input message.

6.3.2.6 The AP-TYPE Component converts the equipment type field of the FP input message. The associated aircraft class is returned.

6.3.2.7 The AP-PACNPC Component converts the pacing/non-pacing airport field of the input message. The airport must be entered.

6.3.2.8 The AP-HRMIN Component converts the hour-minute field (PGTD) of the input message. The hour and minute values are converted to epoch seconds of the current day.

6.3.2.9 The AP-PACING Component converts the arrival pacing airport field of the FP message. A pacing airport must be specified.

6.3.2.10 The AP-ETEVAL Component converts the estimated time enroute field of the input message.

6.3.2.11 The APCMSK Module creates a departure date mask indicating a departure for the current day only.

6.3.2.12 The DB-GTFR Component retrieves flight records corresponding to the entered flight plan. A returned flight record indicates the flight plan already exists.

6.3.2.13 The DB-UPFR Component adds the entered flight plan to the non-OAG file.

6.3.2.14 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.3.2.15 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.3.2.16 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

6.4 The AP-ADDEPT Component adds the actual departure time of a flight to the flight record in the Data Base.

6.4.1 The APDMER Driver Module controls the process of adding the actual departure time to the flight record.

6.4.2 The DB-WAIM Component obtains a work area to be used by the Data Base Subsystem.

6.4.3 The AP-INPUT Component retrieves the DM input message and parses it into message elements.

6.4.4 The AP-ERROR Component outputs an error message if any error is determined by the message processing.

6.4.5 The AP-ACID Component converts the Aircraft Identification field of the DM input message.

6.4.6 The AP-PACNPC Component converts the pacing/non-pacing departure airport field of the input message. The airport must be entered.

6.4.7 The AP-HRMIN Component converts the hour-minute field (PGTD) of the input message. The hour and minute values are converted to epoch seconds of the current day.

6.4.8 The AP-PACING Component converts the arrival pacing airport field of the DM message. A pacing airport must be specified.

6.4.9 The DB-GTFR retrieves the flight record for the specified flight.

6.4.10 The APFVLD Module validates that the retrieved flight has a PGTD close to the entered actual departure time.

6.4.11 The DB-UPFR Component updates the flight record in the Data Base.

6.4.12 The AP-PUTMSG Component issues an accept or reject message after the input message has been validation checked.

6.4.13 The DB-CLUP Component performs any Data Base clean-up required on a normal termination.

6.4.14 The EX-RTDI Component returns control to the Executive after a normal TLM termination.

7.0 The AP-APCOMN Component consists of the three major classifications of components used by more than one Transaction Load Module.

7.1 The AP-CMINPT Component performs verification and conversion to internal form of the control parameters in the transaction message.

7.1.1 The AP-INPUT Component retrieves the parameter table from the Data Base Subsystem, retrieves the transaction message from the Executive Subsystem and parses the transaction message into fields and elements.

7.1.1.1 The APINPT Driver Module drives the process of retrieving the parameter table and the process of retrieving and parsing the transaction message.

7.1.1.2 The AP-GETMSG Component retrieves the transaction message from the Executive Subsystem.

7.1.1.2.1 The APGTMS Driver Module is an interface module which retrieves the transaction message from the Executive Subsystem.

7.1.1.2.2 The EX-AMBI Interface Component passes the transaction message to the Application Subsystem.

7.1.1.3 The DB-TABT Component retrieves the Parameter Table (PT) from the Data Base and stores it in the Application Compool.

7.1.1.4 The AP-PARSM Component parses the transaction message first into fields and then into elements of fields.

7.1.1.4.1 The APARSM Driver Module drives the process of parsing the transaction message into fields and into elements of fields.

7.1.1.4.2 The APARSE Module parses a character string based on a specified delimeter.

7.1.2 The AP-CONVRT Component validates each field in the transaction message and converts each field to internal code.

7.1.2.1 The AP-CTRPC2 Conversion Component validates and converts a center/pacing field.

7.1.2.1.1 The APC2ID Driver Module validates a center/pacing airport code and converts it to internal code. If a center is entered, all pacing airports for that center are retrieved. No default is allowed.

7.1.2.1.2 The DB-TABT Component retrieves internal codes matching the pacing/center code from the Airport Table (AJ) and the Center Table (CP).

7.1.2.2 The AP-STRSTP Conversion Component validates a start/stop field and converts its values to internal binary.

7.1.2.2.1 The APSSTP Driver Module validates the format of the start/stop field and drives its conversion to binary.

7.1.2.2.2 The APNUMC Module converts a numeric character string to internal binary form.

7.1.2.3 The AP-AFFIL Conversion Component parses and validates the affiliation entry and converts each affiliation code to internal form.

7.1.2.3.1 The APAFFL Driver Module parses the affiliation entry and validates and converts the validated codes to internal codes. It also builds the appropriate affiliation header. The affiliation entry consists of either airlines or operational categories but not both.

7.1.2.3.2 The APARSE Module parses the affiliation entry into separate codes based on the delimiter (,).

7.1.2.3.3 The DB-TABT Component retrieves internal airline codes from the Airline Operator Table (AD) and internal operational category codes from the Operational Category Table (OK).

7.1.2.4 The AP-EQUIP Conversion Component parses and validates the equipment entry and converts each equipment code to internal form.

7.1.2.4.1 The APEQUP Driver Module parses the equipment entry and validates and converts the equipment codes to internal form. It also builds the appropriate equipment header line. The equipment codes may be either class or type codes but not both.

7.1.2.4.2 The APARSE Module parses the equipment entry into separate equipment codes based on the delimiter (,).

7.1.2.4.3 The DB-TABT Component retrieves internal equipment class codes from the Class Table (PK) and equipment type codes from the Type Table (PY).

7.1.2.5 The AP-ETECUT Conversion Component validates the estimated time enroute cutoff and converts it to internal binary.

7.1.2.5.1 The APETEC Driver Module validates the estimated time enroute cutoff value and converts it to internal binary.

7.1.2.5.2 The APNUMC Module converts the ETE cutoff to internal binary.

7.1.2.6 The AP-OUTFMT Conversion Component processes the output format element of the transaction message. These output format designators control which report items appear in the output report.

7.1.2.6.1 The APOFPR Driver Module drives the process of validating and converting the output format designator element. It also builds the appropriate title line.

7.1.2.6.2 The DB-TABT Component retrieves the Output Format Table (OX) from the Data Base.

7.1.2.6.3 The APARSE Module parses the output format element into output format designators.

7.1.2.7 The AP-OUTDEV Conversion Component validates and converts the output device entry of the transaction message.

7.1.2.7.1 The APOUPR Driver Module validates the output device field and converts each output device designator to internal code.

7.1.2.7.2 The APARSE Module parses the output device entry into separate output device designators.

7.1.2.7.3 The DB-TABT Component retrieves the Output Driver Table (OD) from the Data Base.

7.1.2.8 The AP-PACALL Conversion Component validates the pacing/all pacing airport field of the transaction message and converts the pacing code(s) to internal code(s).

7.1.2.8.1 The APPACD Driver Module validates a pacing airport code and converts it to internal form. If the pacing airport is defaulted, APPACD provides the internal codes of all pacing airports.

7.1.2.8.2 The DB-TABT Component retrieves the Airport Table (AJ).

7.1.2.9 The AP-CTRPCI Conversion Component validates and converts a center/pacing airport field of a transaction message.

7.1.2.9.1 The APC1ID Driver Module validates the center/pacing airport field of a transaction message and converts it to internal code.

7.1.2.9.2 The DB-TABT Component retrieves entries from the Airport Table (AJ) and the Center Table (CP).

7.1.2.10 The AP-PACING Conversion Component validates and converts a pacing airport code.

7.1.2.10.1 The APPAID Driver Module validates the pacing airport field of a transaction message and converts it to internal code.

7.1.2.10.2 The DB-TABT Component retrieves an entry from the Airport Table (AJ).

7.1.2.11 The AP-DELAYF Conversion Component processes the delay factor elements of the transaction message.

7.1.2.11.1 The APDLYF Driver Module validates the delay factor value.

7.1.2.11.2 The APNUMC Module converts the delay factor integer to internal binary.

7.1.2.12 The AP-HOLDTM Conversion Component processes the hold time element of transaction message.

7.1.2.12.1 The APHOLD Driver Module validates the hold time value.

7.1.2.12.2 The APNUMC Module converts the hold time integer value to internal binary.

7.1.2.13 The AP-STACK Conversion Component processes the stack count and stack time elements of the transaction message.

7.1.2.13.1 The APSTKT Driver Module validates the stack size and the stack time.

7.1.2.13.2 The APNUMC Module converts the stack size to internal binary.

7.1.2.13.3 The AP-HRMINC Conversion Component validates and converts the stack time.

7.1.2.13.3.1 The APHRMC Driver Module converts the stack hour and minute value to seconds from epoch.

7.1.2.13.3.2 The APNUMC Module converts the stack hour and minute to interval binary.

7.1.2.14 The AP-CAPUPD Conversion Component processes the landing capacity update triplets of the transaction message.

7.1.2.14.1 The APCAPP Driver Module drives the process of parsing the update triplets and validates the entered values for start hour, stop hour and landing capacity update value.

7.1.2.14.2 The AP-TABTEI Conversion Component parses a character vector into a repeating group of three simple items and converts each item to internal binary form.

7.1.2.14.2.1 The APTARP Driver Module drives the process of parsing and converting a repeating group of triplets.

7.1.2.14.2.2 The APARSE Module parses the character string into triplets and each triplet into their simple items.

7.1.2.14.2.3 The APNUMC Module converts a character integer to binary.

7.1.2.15 The APGAEUPD Conversion Component processes the General Aviation Estimates update triplets of the transaction message.

7.1.2.15.1 The APGAEP Driver Module drives the process of parsing the GA update triplets and validates the entered value for start hour, stop hour, and GA update value.

7.1.2.15.2 The AP-TABTRI Conversion Component parses a character vector into a repeating group of three simple items and converts each item to internal binary form.

7.1.2.16 The AP-TESTCP Conversion Component processes the test capacity update triplets of the transaction message.

7.1.2.16.1 The APTSTC Driver Module drives the process of parsing the test capacity update triplets and validates the entered start hour, stop hour, and test capacity values.

7.1.2.16.2 The AP-TABTRI Conversion Component parses a character vector into a repeating group of three simple items and converts each item to internal binary form.

7.1.2.17 The AP-ZONEID Conversion Component validates the zone entered in the transaction message and converts it to internal binary.

7.1.2.17.1 The APZONE Driver Module drives the process of validating and converting the zone identifier.

7.1.2.17.2 The APNUMC Module converts the zone identifier to internal binary.

7.1.2.17.3 The DB-TABT Component retrieves entries from the Zone Table (ZO).

7.1.2.18 The AP-DEPDLY Conversion Component parses, validates and converts the departure delay pairs of the transaction message.

7.1.2.18.1 The APDDLE Driver Module drives the parse, validation, and conversion of the departure delay pairs.

7.1.2.18.2 The APARSE Module parses the departure delay element into pairs and further parses each pair into delay factor and airport code.

7.1.2.18.3 The APNUMC Module converts the delay factor integer to internal binary.

7.1.2.18.4 The DB-TABT Component retrieves entries from the Airport Table (AJ).

7.1.2.19 The AP-PACNPC Conversion Component processes the pacing/non-pacing airport code of the transaction message.

7.1.2.19.1 The APPANP Driver Module validates and converts the pacing/non-pacing airport code and retrieves the associated internal center code.

7.1.2.19.2 The DB-TABT Component retrieves entries from the Airport Table (AJ) and from the Center Table (CP).

7.1.2.20 The AP-CTRAPT Conversion Component processes the center/pacing airport/non-pacing airport code of the transaction message.

7.1.2.20.1 The APCPNI Driver Module validates and converts the center/pacing airport/non-pacing airport code.

7.1.2.20.2 The DB-TABT Component retrieves entries from the Airport Table (AJ) and the Center Table (CP).

7.1.2.21 The AP-ACID Conversion Component processes the aircraft identification field of the transaction message.

7.1.2.21.1 The APACID Driver Module validates the aircraft identification and converts the airline operator to

internal code. It also retrieves the appropriate operational category code.

7.1.2.21.2 The DB-TABT Component retrieves entries from the Airline Operator Table (AO), the non-OAG Name Table (NO), and the Operational Category Table (OK).

7.1.2.22 The AP-ACIDSG Conversion Component processes the aircraft identification/airline operator code of the transaction message.

7.1.2.22.1 The APACAL Driver Module validates the aircraft identification/airline operator code and converts the airline operator to internal code.

7.1.2.22.2 The DB-TABT Component retrieves entries from the Airline Operator Table (AO).

7.1.2.23 The AP-FIXPAC Conversion Component processes the Fix/Pacing Airport code of the transaction message.

7.1.2.23.1 The APFPID Driver Module validates the Arrival Fix/Pacing Airport code, converts the code to internal code, and retrieves the associated Pacing Airport/Arrival Fix.

7.1.2.23.2 The DB-TABT Component retrieves entries from the Airport Table (AJ), the Arrival Fix Table (AV), and the Airport Fix Table (AF).

7.1.2.24 The AP-TYPE Conversion Component processes the aircraft equipment type code.

7.1.2.24.1 The APTYPE Driver Module validates the aircraft equipment type code and converts it to internal code. It also retrieves the associated class code.

7.1.2.24.2 The DB-TABT Component retrieves entries from the Type Table (PY) and the Class Table (PK).

7.1.2.25 The AP-ETEVAL Conversion Component processes the estimated time enroute value of the transaction message.

7.1.2.25.1 The APETEV Driver Module validates the estimated time enroute.

7.1.2.25.2 The APNUMC Module converts the ETE integer value to internal binary.

7.1.2.26 The AP-HRMIN Conversion Component processes an hour-minute element of the transaction message.

7.1.2.26.1 The APTIHM Module validates an hour-minute character vector and converts it to epoch seconds.

7.1.2.26.2 The AP-NRMINC Conversion Component converts an hour-minute character vector to epoch seconds.

7.1.2.26.2.1 The APHRMC Driver Module converts an hour-minute character vector to epoch seconds.

7.1.2.26.2.2 The APNUMC Module converts an hour value and minute value to binary.

7.1.2.27 The AP-DATE Conversion Component processes any Julian or Gregorian date character vector.

7.1.2.27.1 The APDATE Driver Module validates and converts a date character vector to epoch seconds.

7.1.2.27.2 The APETIM Module computes current date in MMDDYY from epoch transaction time.

7.1.2.27.3 The AP-CONJUL Component processes a Julian date.

7.1.2.27.3.1 The APCVJL Driver Module validates and converts a Julian date to elapsed seconds from epoch.

7.1.2.27.3.2 The APITIM Module converts a calendar date to elapsed seconds from epoch.

7.1.2.27.3.3 The APNUMC Module converts a Julian date in character form to internal binary.

7.1.2.27.4 The AP-CONGRG Component processes a Gregorian Date.

7.1.2.27.4.1 The APCVMD Driver Module validates and converts a Gregorian date to epoch seconds.

7.1.2.27.4.2 The APNUMC Module converts the Gregorian date character vector to internal binary.

7.1.2.27.4.3 The APITIM Module converts the binary Gregorian date to elapsed seconds from epoch.

7.1.2.27.4.4 The APETIM Module converts epoch seconds to a calendar date format.

7.1.2.28 The AP-SSDATE Conversion Component processes a Start/Stop date.

7.1.2.28.1 The APSSDA Driver Module validates and converts a Start/Stop date character vector.

7.1.2.28.2 The APETIM Module computes current date in MMDDYY from epoch transaction time.

7.1.2.28.3 The AP-CONJUL Component processes a Julian date.

7.1.2.28.4 The AP-CONGRG Component processes a Gregorian date.

7.1.2.29 The AP-SSDTTM Conversion Component processes a start-stop date/time element of a transaction message.

7.1.2.29.1 The APSSDT Driver Module validates a start-stop date/time pair and converts the pair to epoch start/ stop seconds.

7.1.2.29.2 THe APETIM Module computes current Gregorian date from transaction time.

7.1.2.29.3 The AP-CONJUL Component processes a Julian date.

7.1.2.29.4 The AP-CONGRG Component processes a Gregorian date.

7.2 The AP-CMTRFM Common Processing Component consists of all components and modules common to more than one transaction load module whose function is to manipulate data to create the desired report.

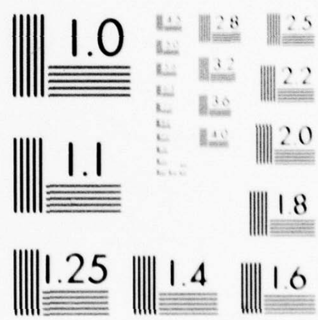
7.2.1 The AP-BLDTRF Component builds a time-ordered list of traffic.

7.2.1.1 The APBLDT Driver Module builds a time-ordered list of arrival or departure flights.

AD-A070 973 COMPUTER SCIENCES CORP SILVER SPRING MD SYSTEM SCIEN--ETC F/G 9/2
CENTRAL FLOW CONTROL SOFTWARE DESIGN DOCUMENT. VOLUME I. OPERAT--ETC(U)
JAN 79 DOT-FA77WA-3955
UNCLASSIFIED CSC/SD-78/6172-1 FAA-RD-79-33-1 NL

6 OF 6
AD A070973

END
DATE
FILMED
8-79
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

7.2.1.2 The DB-TABT Component retrieves entries from the Airport Table (AJ).

7.2.1.3 The APSORT Module sorts the traffic list in ascending order based on arrival/departure times.

7.2.2 The AP-DEMBLD Component builds demand counts of arrivals or departures for today.

7.2.2.1 The APDBLD Driver Module builds a report of hourly arrival/departure counts. It also provides a count of active arrival/departure flights and, if requested, the hourly GA count.

7.2.2.2 The DB-TABT Component retrieves entries from the User GA Table (UJ).

7.2.3 The AP-INTCNT Component builds and initializes demand counters.

7.2.3.1 The APBCNT Driver Module creates a future demand count by hourly increments.

7.2.3.2 The APETIM Module converts epoch start time into start hour and start date.

7.2.3.3 The APCONV converts the start date to EBCDIC.

7.2.4 The AP-FLPBLD Component builds a list of flight plans.

7.2.4.1 The APBLDF Driver Module builds a time-ordered list of flight plans.

7.2.4.2 The APSORT Module sorts the flight plans in ascending order based on proposed time of arrival.

7.2.5 The AP-BLDFUT Component builds future demand counts.

7.2.5.1 The APDBLF Driver Module builds a report of future demand counts of arrivals or departures. GA adjustments may be made.

7.2.5.2 The APFVLD Module validates that a given flight arrives or departs within a specified report period and computes the epoch arrival/ departure times.

7.3 The AP-CMOUPT Component consists of all common output processing.

7.3.1 The AP-ERROR Component processes an error message.

7.3.1.1 The APEROR Driver Module issues an error message and, if the error is a critical error, calls the Executive Subsystem to abend.

7.3.1.2 The AP-PUTMSG Component prepares and issues an immediate message.

7.3.1.2.1 The APPTMS Driver Module prepares and issues an immediate message.

7.3.1.2.2 The DB-TABT Component retrieves the appropriate error message from the Message Table (MM).

7.3.1.2.3 The AP-SENDIM Component constructs and sends an immediate message block to the Executive Subsystem for output.

7.3.1.2.3.1 The APSNIM Driver Module builds the message to be output.

7.3.1.2.3.2 The EX-SMBI Interface Component issues the immediate message.

7.3.1.2.4 The EX-GDRI Interface Component aborts transaction load module processing when a system error condition has been detected.

7.3.1.3 The EX-GDRI Interface Component aborts transaction load module processing when a system error condition has been detected.

7.3.1.4 The DB-CLUP Component performs all clean-up operations following the completion of each transaction load module.

7.3.2 The AP-OUTHDR Component formats and sends to the Executive Subsystem the specified report header lines.

7.3.2.1 The APOHDR Driver Module builds and sends report header lines and report title line to the Executive Subsystem.

7.3.2.2 The AP-FORMAT Component formats the output line by using either the header format table or the data format table as a guide.

7.3.2.2.1 The APFRMT Driver Module formats the output line.

7.3.2.2.2 The DB-TABT Component retrieves external codes required for the output header lines or data lines.

7.3.2.2.3 The APETIM Module converts epoch time to calendar format for output.

7.3.2.2.4 The APCONV converts binary numbers to character form for output.

7.3.2.3 The AP-SENDRP Component outputs each line of data to the Executive Subsystem.

7.3.2.3.1 The APSNRP Driver Module controls the procedure to output report lines.

7.3.2.3.2 The EX-SMBI Component is the Executive Interface which issues the message block.

7.3.3 The AP-OUTDAT Component formats and sends to the Executive Subsystem the specified report data lines.

7.3.3.1 The APODAT Driver Module builds and sends report data lines to the Executive Subsystem.

7.3.3.2 The AP-FORMAT Component formats the output line by using either the header format table or the data format table as a guide.

7.3.3.3 The AP-SENDRP Component outputs each line of data to the Executive Subsystem.

7.3.4 The AP-QFLWRP Component formats and issues the data section the quota flow report.

7.3.4.1 The APQFRP Driver Module controls the process of building and issuing the quota flow first tier report.

7.3.4.2 The APCONV Module converts binary numbers to character form for the output report.

7.3.4.3 The AP-OUTDAT Component formats and sends to the Executive Subsystem the specified report data lines.

7.3.5 The AP-UPDCNT Component updates the Simulation Continue Table (CO).

7.3.5.1 The APCTUP Driver Module controls the process of updating information in the Simulation Continue Table (CO).

7.3.5.2 The DB-SETT Component inserts the updated values in the Continue Table (CO).

7.3.6 The AP-UPSIM Component updates flight records as a result of Simulation processing.

7.3.6.1 The APSUPD Driver Module controls the process of updating flight records as indicated by the Simulation components.

7.3.6.2 The DB-UPFR Component updates a flight record in the Data Base.

2.2.1.5 Simulation (SI) Subsystem

a. Purpose. The purpose of the Simulation (SI) Subsystem is to provide airport airborne arrival delay predictions and to compute flow control tables for air traffic management of impacted airports. Simulations are performed at the request of the Air Traffic Control System Command Center (ATCSCC).

b. Subsystem Overview. An Arrival Delay Prediction (ARRD) and two flow control procedures (QFLOW, FAD) are supported. The Simulation Subsystem works in conjunction with input from the Application Subsystem (AP), centralized data management services from the Data Base Management Subsystem, and output components in the Application Subsystem to generate and transmit reports. Figure 2.2.1-14 presents the Simulation Subsystem Overview.

Given input message data from AP, the requested ARRD, QFLOW, OR FAD simulation first requests, in a related setup component, necessary table data and a set of qualified flight records from the Data Base. A process component is then invoked to perform the requested simulation from which output report tables, a list of controlled flights, and/or error messages are prepared. The Simulation Subsystem is designed to meet the requirements as specified in the document titled: "A Computer Program Functional Design of the Simulation Subsystem of an Automated Central Flow Control System", FAA Report No. FAA-RD-76-144.

c. Functional Description.

1. ARRD Simulation

Objective. The objective of the ARRD simulation is to predict arrival delays at a specified pacing airport. Reports are generated based on OAG flight

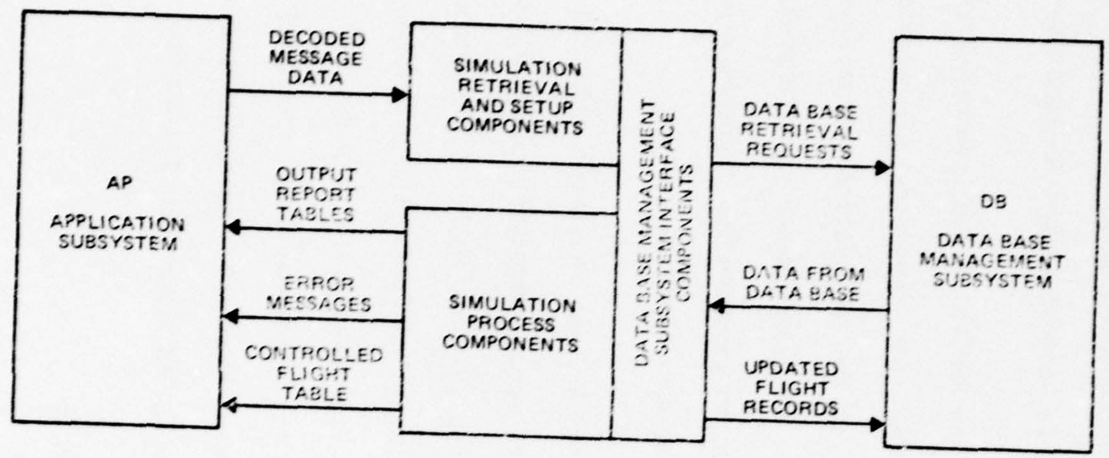


Figure 2.2.1-14. SI-1.0 Simulation Subsystem (SI) Interface Overview

schedules, real-time ARTCC-originated updates, hourly general aviation factors, known stack conditions at some specified time, and hourly landing rates for the destination airport. Each report includes the following information:

- A count of the number of aircraft expected to arrive at the destination airport during each hour of the report.
- A count of the arrivals for which an ARTCC departure message (DM) has been received.
- Based on the hourly landing capacity, a count of the predicted number of aircraft landed during each hour of the report.
- The peak and average number of aircraft predicted to be holding in the terminal area for each hour of the report.
- Peak and average delay predicted for any aircraft holding during each report hour.

Methodology. The ARRD simulation methodology consists of stepping through a prepared event list, one event at a time, and pausing to perform the stack adjust, if required. A landing time is computed for the flight based on specified landing rates for the airport being simulated. Arrival delay data is gathered for each arrival flight event. Finally, after sequencing through all of the flights in the list, the data collected is summed and averaged to produce the hourly statistics.

In order to collect the counts needed to determine the peak and average number of aircraft holding during each hour, a set of sixty counters, corresponding to

the minutes of the hour, are provided for each hour of the report. In the event that the arriving flight must hold, i.e., the next available landing slot is later than the arrival time, every minute counter that corresponds to the time-span bounded by and including the arrival time up to the landing time will be incremented. The counters corresponding to respective hours will be summed and averaged. The maximum of these counts will be retained as the peak number of aircraft holding for that hour. To retain the overall peak delay experienced by arrival aircraft during each hour, the delay assigned to the first arrival in the hour is retained as the initial peak delay. The delay assigned to each succeeding arrival is then compared to the existing peak delay and the larger value is retained as the peak delay for the arrival hour.

Cumulative total delays for each hour will be kept by adding each arrival flight's delay to the cumulative for that arrival hour. The average hourly delay is then determined by dividing the hourly cumulative by the number of delays added during each hour. If the stack adjust time is encountered, the existing number of holding aircraft must be adjusted up or down to conform to this size. This will improve the delay prediction accuracy for the remainder of the report. To accomplish this, the program resets all holding counters and accumulators associated with times later than this stack adjustment time. All landing slots later than this time are released for reassignments. A number of previously landed aircraft are then brought back into the stack, or a number of aircraft are immediately dequeued from the stack to force the stack size to the input specification. The processing of real arrival flights will then continue. Processing is completed when the event list has been exhausted. The delay statistics that have been recorded are passed to the message output component for formatting and transmittal to the Executive Subsystem.

2. QFLOW Simulation

Objective. The objective of the QFLOW simulation is to compute tier center release rates, carryover counts, and other subsidiary information which will be used to limit the peak delay in the terminal area to some controller-specified level. Information on which this report is based includes that described above for ARRD, a zone structure that relates flight origins to tier centers, a maximum delay value to be imposed for terminal area holding time, and the earliest time at which these computed release rates can be implemented (First Notice Time). Each report includes the following information for each hour covered:

- The number of aircraft to be released from each tier center during this hour that will maintain a constant demand on the airport based on the maximum hold time specified for the terminal area.
- The number of flights that must be detained in the tier center at the end of this hour (tier center carryover).
- The number of general aviation flights that can be expected to contribute to each hourly interval. This release rate includes only the general aviation (GA) traffic created by the simulation from estimated hourly GA counts for the airport. Since these dummy flights have no origin, no attempt is made to associate them with specific tier centers. They will appear as an aggregate to be added to the total release rate for all tier centers.
- For the quota flow by zone simulation (QFLZ), the report will also include a list of associated origins for each tier and the number of releases each origin contributes to

the total release rate for the tier.

Methodology. The QFLOW simulation methodology consists of the same sequence of processing events and assigning landing slots as outlined for ARRD.

In order that tier center release rates be properly applied, a zone structure must be available to the simulation process. This structure relates departure centers to the first tier centers of the destination airport; i.e., the zone structure determines the path an arrival flight will take from the departure airport or center through a particular tier center to the destination airport. This structure can then be used to determine in which tier center a flight should be delayed and subsequently released.

There may be several of these structures for a destination airport to provide for changes in traffic routing during the day. The simulation controller will select one of these structures by specifying a structure code in the input message. Additional items available in the zone structure must include a nominal flight time for general aviation dummy flights created for the simulation, so that they may be placed in an approximately correct release interval, and a boundary flight time for the tier center; i.e., the nominal flight time from the tier center boundary to the destination airport fix.

One final piece of information required for the QFLOW simulation process is an Earliest Notice Time. This is defined as the earliest time that the flow controls resulting from this simulation could be implemented, and is considered to be the simulation message receipt time plus a communication lag time constant.

As arrival events are selected from the events list, a delay for the flight is

computed by subtracting its assigned landing time from the estimated time of arrival. If this delay exceeds the controller-specified maximum delay, an arrival time is assigned that is equal to the landing slot time minus the controller maximum delay time (provided Earliest Notice Time is not violated). After the arrival time assignment is made, the time interval in which this flight should be released from its tier center is computed by subtracting the tier center boundary flight time from the arrival time. A release rate counter associated with this time interval and tier center is then incremented. To determine the number of flights to be held over in the tier center for each release interval, the time of tier center boundary arrival is computed from the departure time plus estimated time enroute,minus boundary flight time. If this boundary arrival time hour is not the same as the release time hour interval, carryover counters for each intervening hourly interval for this tier center will be incremented. This process maintains the carryover counts. If the simulation is a quota flow by zone (QFLZ), an origin counter associated with this flight's origin and tier center is incremented for the appropriate interval to provide origin counts.

If the arrival event being processed has nonzero controlled arrival and departure time fields, and the continue simulation flag is set indicating flow controls from a previous simulation, it will be necessary to examine these times to determine whether all or part of the previously assigned controls can now be removed to take full advantage of the new simulation. This decision is based on the Earliest Notice Time, described above, and the relationship of this time to the controlled departure and arrival times in the arrival flight event record. To determine if ground delay can now be removed, the flight's planned departure time is compared to Earliest Notice Time. If the planned departure time is the later time, the

flight will have taken no ground delay time at Earliest Notice Time and therefore, controlled departure time and best estimate of arrival time can be reset to the original planned times prior to applying the present simulation. If planned departure time is earlier than Earliest Notice Time and controlled departure time is later than Earliest Notice Time, the flight has taken part of its ground delay time, provided an actual departure message has not been received. In this case, controlled departure time may be reset to Earliest Notice Time and the best estimate of arrival time reset to original planned arrival time minus the delay taken on the ground up to Earliest Notice Time. If the flight's controlled departure time is earlier than Earliest Notice Time, its previously assigned delay cannot be altered and the controlled times remain as is for the present simulation.

A comparison of the flight's tier center release time versus Earliest Notice Time is now made. If the tier center release time is the earlier time, the flight's arrival time is unchangeable at this point. If the tier center arrival time is later than Earliest Notice Time and it had been assigned a tier center delay, then its arrival time will be reset to boundary arrival time plus boundary flight time plus the delay taken at the tier center up to Earliest Notice Time.

If the next arrival event receives a landing time later than stack adjust time, the stack size is adjusted as described above. Processing will then continue with the next arrival event. Processing stops when the event list has been exhausted. The simulation statistics that have been accumulated are passed to the message output component for formatting and transmittal to the Executive.

3. FAD Simulation

Objective. The objective of the FAD simulation is to compute control delays, tier center release rates, and subsidiary information which will be used to limit the peak delay in the terminal area to some controller-specified level. To present the statistics that would result from the implementation of the control delays and tier center release rates, an ARRD simulation is computed indicating the delay parameters resulting from the imposed delays. This report is based on information already described under the ARRD and QFLOW plus controller-specified ground delays for certain terminals (FADT only), and an ETE cutoff time which is used to restrict ground delays to flights with enroute time less than this ETE value. The FAD simulation generates several types of reports, depending on the type of FAD simulation requested.

1. FADF

- A listing in hourly increments of the landing capacities (CAPL) for the specified pacing airport covering the report period.
- A listing of general aviation estimates (GAEL) for the specified pacing airport for the reporting period.
- If a controller input zone qualifier is specified, a QFLOW report for the pacing airport for the reporting period.
- An ARRD report indicating the controlled arrival delay predictions for the specified airport.

- A flight list report for individual flights, listing the assigned delays as a result of this simulation and including the departure airport, estimated departure clearance time, proposed gate departure time, ground assigned delays, total delay, ETE, new estimated time of arrival, and boundary crossing time.

2. FADP. This report includes the first four items from the FADF report. The last item in this report is the block delay list. The list contains the assigned delay times in 15-minute increments covering the report period.

3. FADT. This report includes items 3 and 4 from the FADF report and the block delay report from FADP. Test capacity values for the report period are utilized in assembling the reports and simulations. These test values are input by the controller, as opposed to other FAD simulations where the stored data base capacities are utilized, i.e., this simulation is essentially a test run to determine if FAD controls are necessary.

Methodology. The FAD simulation methodology consists of the same sequence of processing events and assigning landing slots as outlined for the ARRD simulation.

Tier center release rate computations for the FAD simulation are identical to those described for the QFLOW simulation except that for FADT simulations, they are based on delayed departure times from the message input, where applicable. The process of examining event records for possible removal of

previously assigned delays is the same as described for the QFLOW simulation and needs no further elaboration.

Arrival delay statistics computed in the FAD simulation involve the same processes as described for the ARRD simulation except that the terminal area arrival times used in the computation are controlled arrival times assigned by the FAD simulation.

If the next event is an arrival flight event, the simulation process computes a controlled arrival time equal to the assigned landing time minus the maximum desired delay at the destination airport for those flights whose delay would exceed this specified maximum. If the flight's estimated time enroute is less than the controller-specified ETE cutoff, and it has not departed or will not depart before first notice time, its departure is delayed by an amount equal to its controlled arrival time minus its best estimate of arrival time, provided Earliest Notice Time is not violated. After each arrival time assignment, a cumulative total of control delays for the interval associated with the flight's planned time of arrival is increased by the amount of this flight's control delay i.e., controlled arrival time minus planned arrival time. To derive the average control delay from this total, a count of flights contributing to this total is kept. To determine the average terminal area delay, a cumulative total of each control delayed flight's terminal area delay is maintained along with a count of flights contributing to it. After the flight event list is exhausted, the cumulative totals for each interval are divided by the respective interval counts to determine the average control delays to be applied to flights that were scheduled to arrive during these intervals and to determine the average terminal area delays that are projected to be experienced by the same aircraft.

The total average delay for each interval is then determined as the sum of the ground delay and average terminal area delay for each interval of the report.

Stack adjust processing is identical to that described for the QFLOW simulation. FAD simulation is completed when the event list has been exhausted. The delay and flow tables that have been assembled are then passed to the message output component for formatting and transmittal to the Executive Subsystem.

The flight event list to be used as part of the input to the simulation processing consists of an OAG flight list merged with real general aviation flights and a dummy flight list. The entire list is ordered by planned arrival time.

The preparation of the flight event list is accomplished by separately assembling the OAG and non-OAG flights that comprise the flight list. The non-OAG portion is then scanned and hourly counts of real general aviation aircraft are determined. If this is a continued simulation, each new real GA flight is assigned a delay based on average delays from the previous simulation. The number of general aviation dummy flights to be distributed in the segment is then determined by the associated hour's general aviation count compared with the total from the scan. The difference will be the number of dummy flights to be added. Additional flight event list preparation involves adding any zone information to the flight record, computing best estimates of arrival and departure, and computing missing items in the event entries.

The zone information stored in the flight event record consists of a numeric subzone index, determined by the departure center, a nominal flight time for dummy GA flights, and a boundary time representing the flying time from the subzone boundary to the arrival airport. The subzone index is used to determine

through what subzones (tier centers) the departed aircraft will pass in traveling to its destination. The boundary time is used to compute the subzone boundary arrival time and boundary release time during simulation processing.

A stack adjustment event involves adjusting the number of aircraft holding to conform with a user-input stack count. Previously landed aircraft are either brought back into the stack or a number of aircraft are dequeued from the stack depending on whether the computed stack size must be increased or decreased. The stack adjustment is triggered by the first flight processed that is assigned a landing time later than the stack adjustment time.

Landing time computations are performed on aircraft due for arrival within the message-specified report period. The computation of a landing time is influenced basically by three factors: the arrival time of the aircraft, the hourly landing capacity, and the time at which a previous aircraft landed. A preliminary computation is needed to determine the minimum time interval required between landing times. A service time for an aircraft landing in a given hour is computed for each hour of the simulation period by dividing the hour by the landing capacity for that hour.

The arrival time is first verified to be in a nonzero capacity hour and, if no delays are involved, it is assigned a landing time equal to either its arrival time or the beginning of the first nonzero capacity hour occurring after its arrival time. If a delay is involved, that is, if the arrival time is earlier than the last landing time plus the computed time interval between landings, further checks must be made. If the last landing time plus the

interval between landings does not extend into the next hour, this value is assigned as the landing time. If the value extends into the next hour, and it is a nonzero capacity hour, the landing time is computed as the lesser of the last landing time plus its service interval, or the beginning of the subsequent hour plus its service interval. Otherwise, the landing time is computed as the beginning of the next nonzero capacity hour.

d. Major Components. The Simulation Subsystem is logically divided into three major components.

- The Arrival Delay Predictions Component (SI-ARRD) performs the ARRD simulation.
- The Quota Flow Simulation Component (SI-QFLOW) performs the QFLW and QFLZ simulations.
- The Fuel Advisory Simulation Component (SI-FAD) performs the FADF, FADP, and FADT simulations.

e. Input. The inputs to the SI Subsystem are:

- Data derived from the simulation input message
- OAG and non-OAG flight records
- Environmental tables (Data Base)

f. Output. The output of the SI Subsystem consists of:

- Simulation output reports
- Flight record updates reflecting flow controls from a FADF or FADF simulation.
- Updates to the Continue Table (except for ARRD and FADT).

2.2.1.5.1 Arrival Delay Predictions (SI-ARRD) Component

a. Purpose. The purpose of the SI-ARRD Component is to predict arrival delays at a specified pacing airport.

b. Functional Description. The SI-ARRD Component retrieves table data from the Data Base for the ARRD simulation using the SI-TABRET Component. The SI-SETUPA Component then retrieves flight records and creates the event list, accounting for GA and dummy flights. The event list is ordered by planned arrival time and is used by the SI-PRCSA Component to perform the ARRD simulation.

c. Input. The inputs to the SI-ARRD Component are:

- Derived data from the ARRD input message
- Flight records
- Environmental tables

d. Output. The output of the SI-ARRD Component consists of:

- An ARRD report table.

2.2.1.5.2 Quota Flow Simulation (SI-QFLOW) Component

a. Purpose. The purpose of the SI-QFLOW Component is to compute tier center release rates and carry over counts to limit the peak delay in the arrival terminal area to a controller-specified level.

b. Functional Description. The SI-QFLOW Component retrieves table data from data base for either the QFLW or QFLZ simulation using the SI-TABRET Component. The SI-SETUP Component then retrieves flight records and creates the event

list, accounting for GA and dummy flights. The event list is ordered by planned arrival time and is used by the SI-PRCSQ Component to perform the QFLW or QFLZ simulation, whichever is requested. Update information for the Simulation Continue Table is prepared.

c. Input. The inputs to the SI-QFLOW Component are:

- Derived data from the QFLW or QFLZ input message
- Flight records
- Environmental tables
- Simulation Continue Table

d. Output. The output of the SI-QFLOW Component consist of:

- QFLOW and ARRD simulation reports
- Updates to the Simulation Continue Table

2.2.1.5.3 Fuel Advisory Simulation (SI-FAD)

a. Purpose. The purpose of the SI-FAD Component is to compute flow control delays and tier center release rates to limit the peak delay in the arrival terminal area to a controller-specified level.

b. Functional Description. The SI-FAD Component retrieves table data from the data base for either the FADF, FADP, or FADT simulation using the SI-TABRET Component. The SI-SETUPF Component then retrieves flight records and creates the event list, accounting for GA and dummy flights. The event list is ordered by planned arrival time and is used by the SI-PRCSF Component to perform a FAD simulation. Update information for the Simulation Continue Table is prepared when processing a FADF or FADP message.

c. Input. The inputs to the SI-FAD Component are:

- Derived data from the FADF, FADP, or FADT input message
- Flight records
- Simulation Continue Table
- Environmental Tables

d. Output. The output of the SI-FAD Component consist of:

- FAD, QFLOW, and ARRD simulation reports
- Updates to the Simulation Continue Table (FADF and FADP)
- Updates to those flight records assigned flow controls (FADF and FAD).

The Visual Table of Contents for the SI Subsystem is shown in Figure 2.2.1-15.

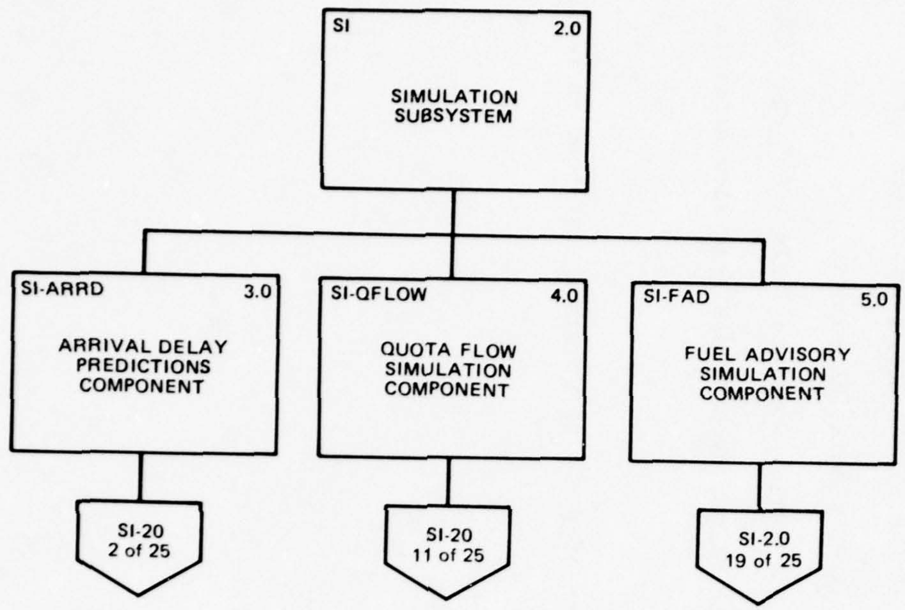


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (1 of 25)

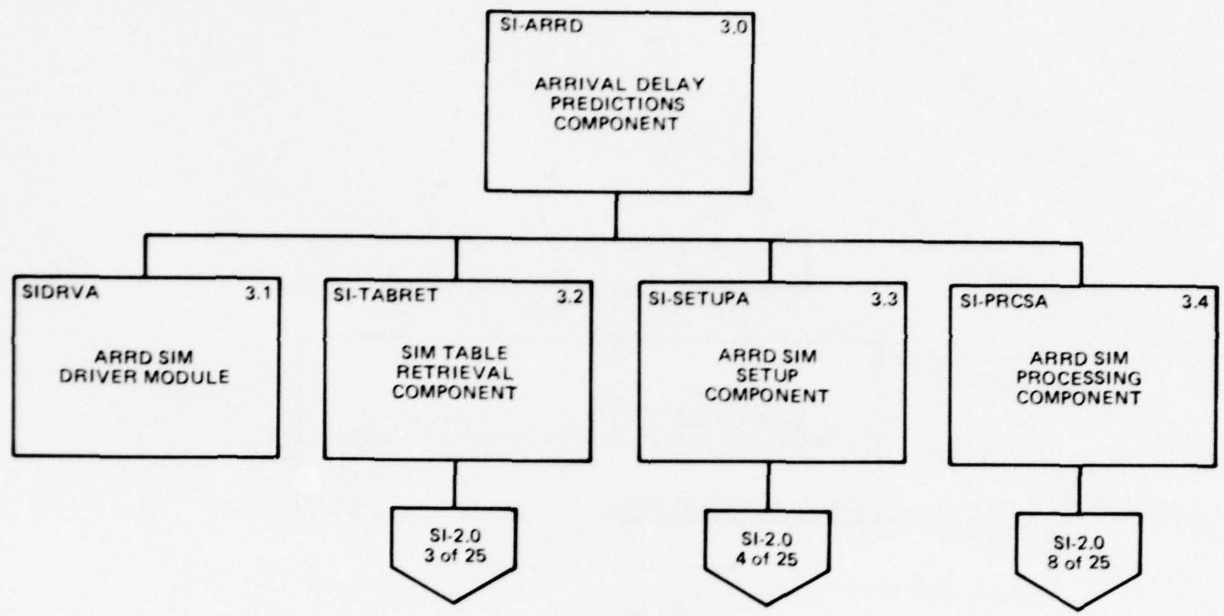


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (2 of 25)

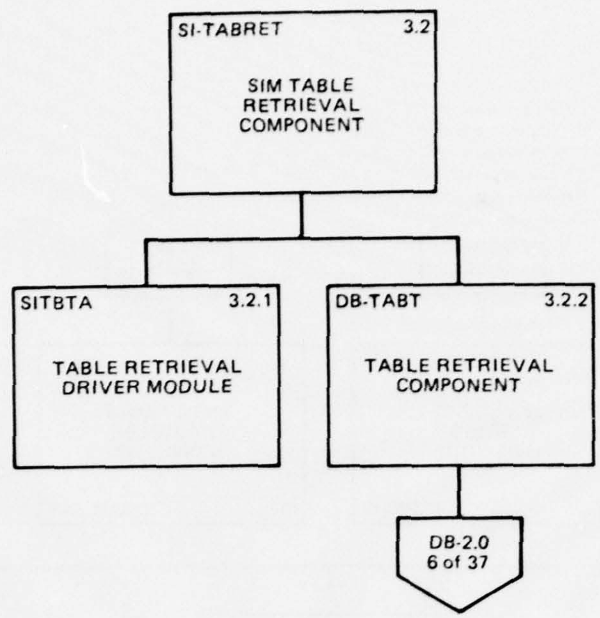


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (3 of 25)

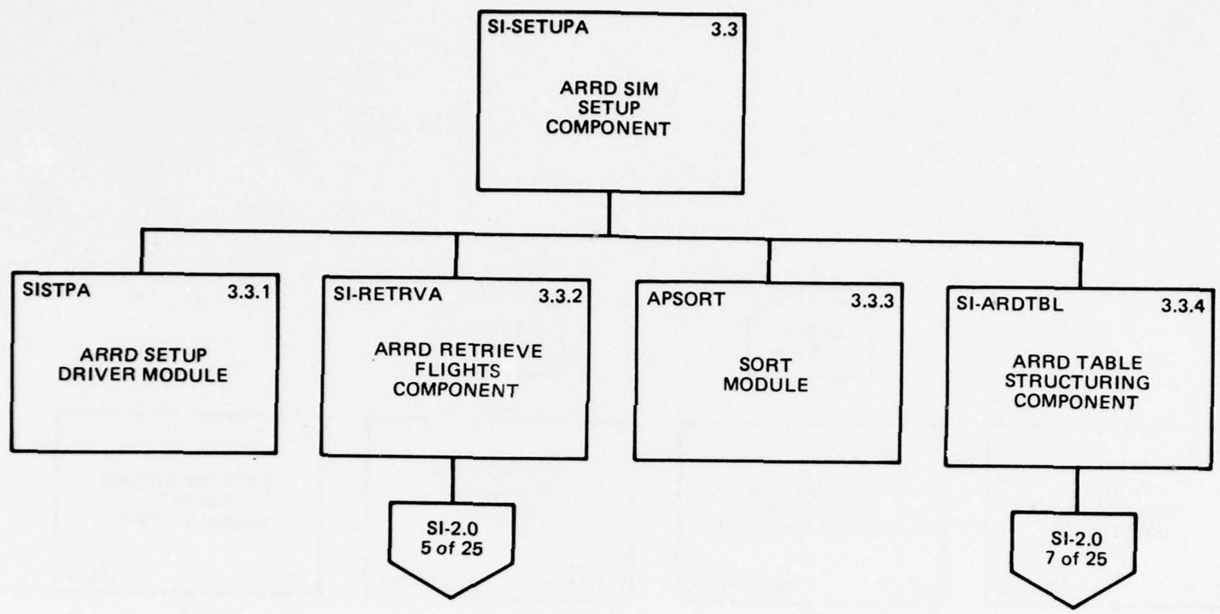


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (4 of 25)

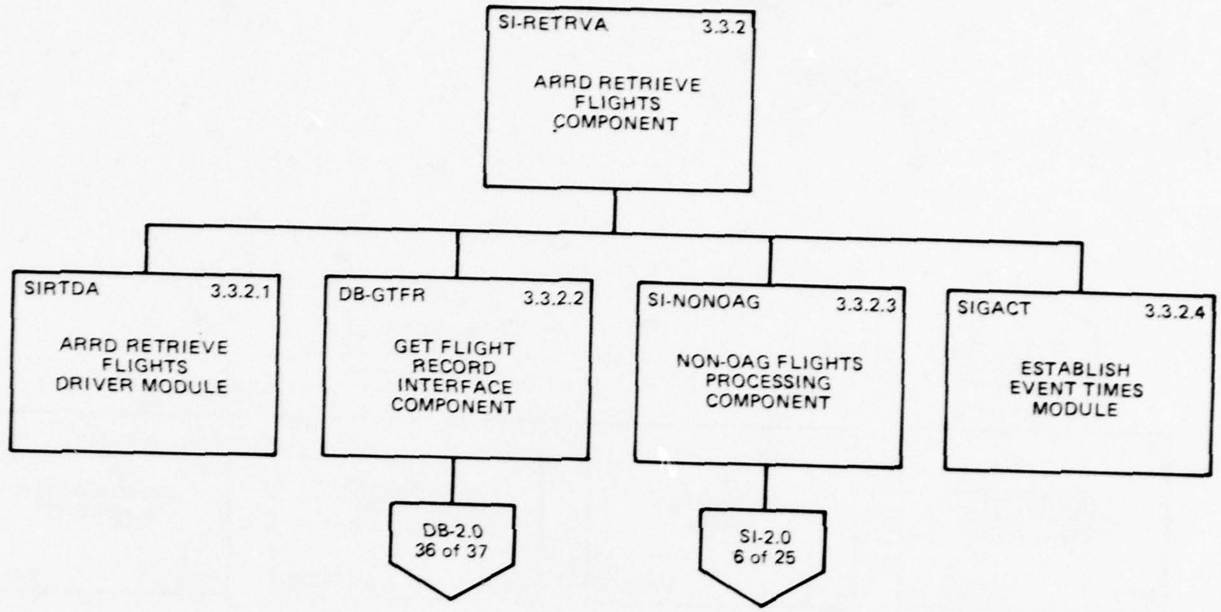


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (5 of 25)

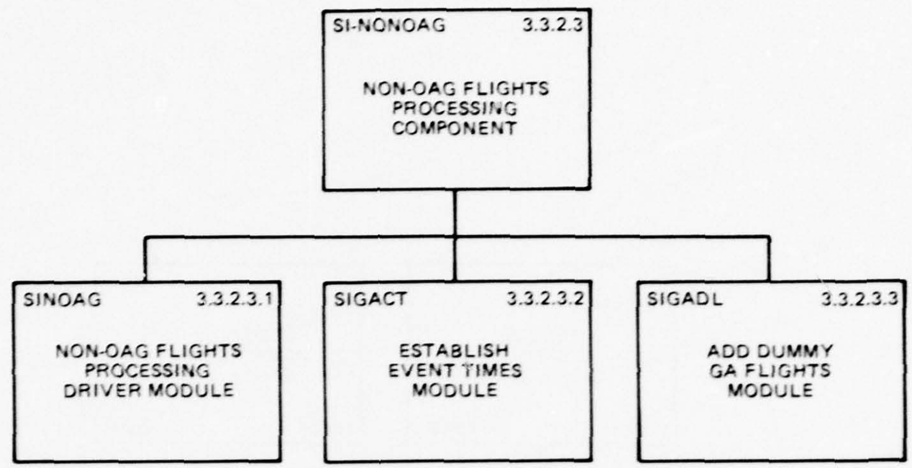


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (6 of 25)

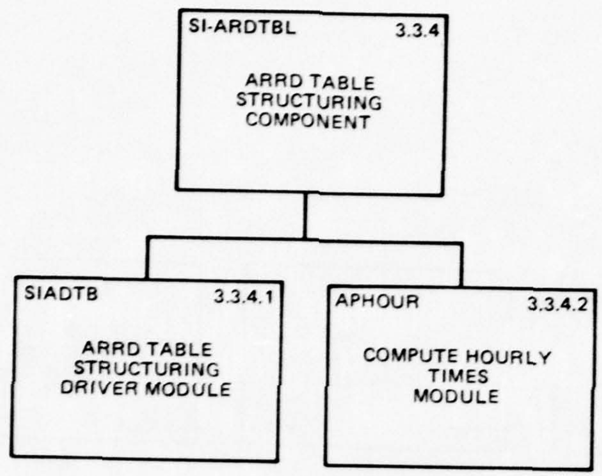


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (7 of 25)

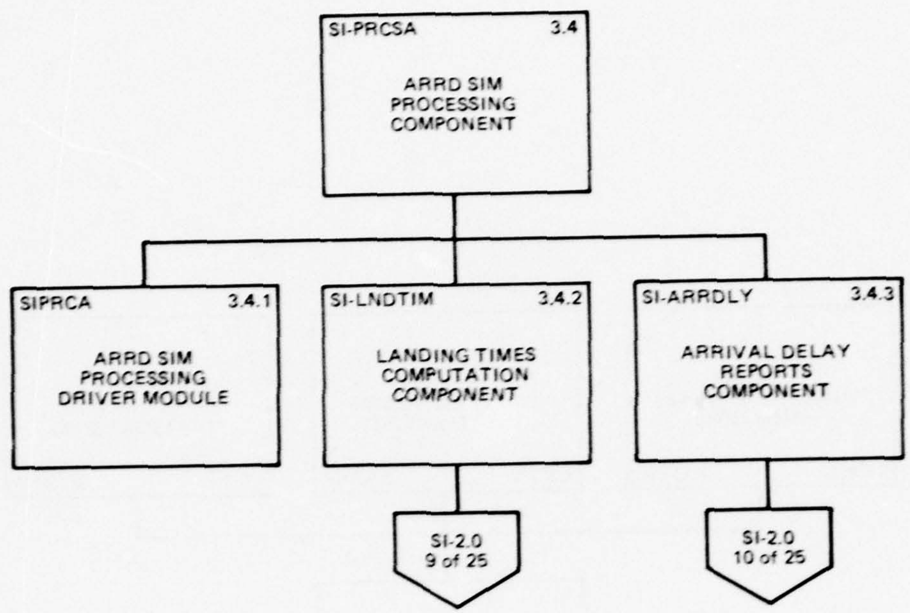


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (8 of 25)

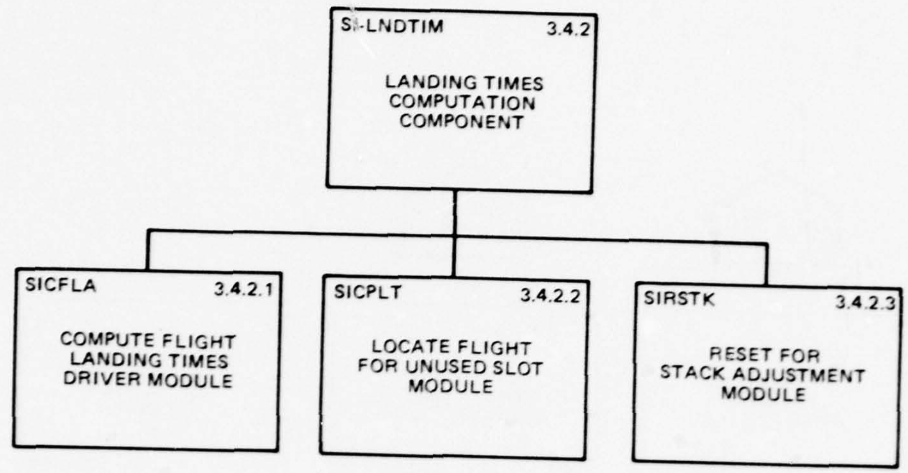


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (9 of 25)

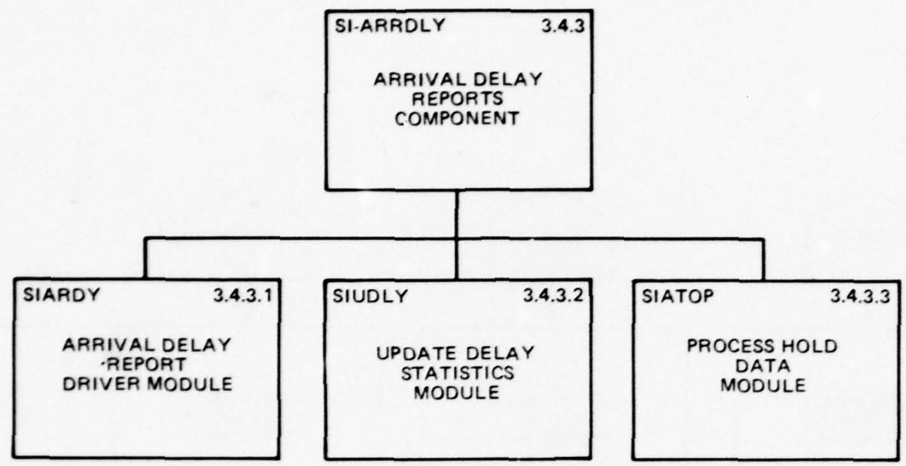


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (10 of 25)

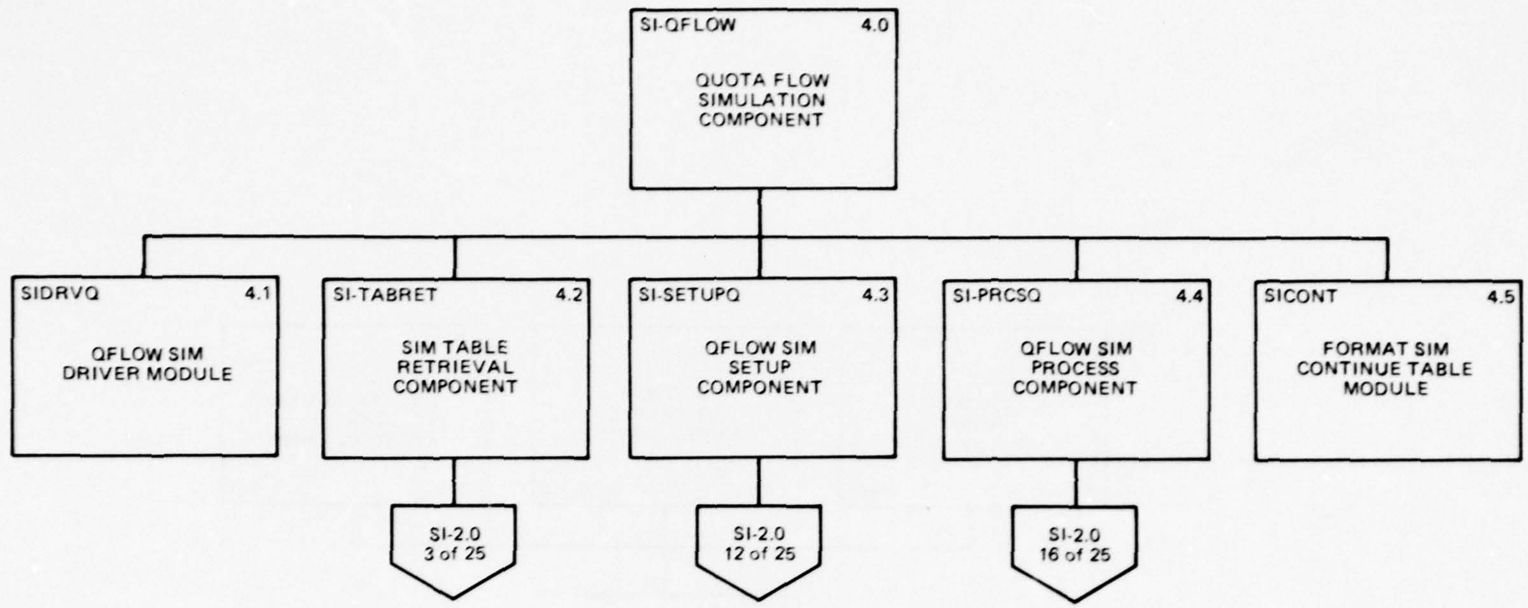


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (11 of 25)

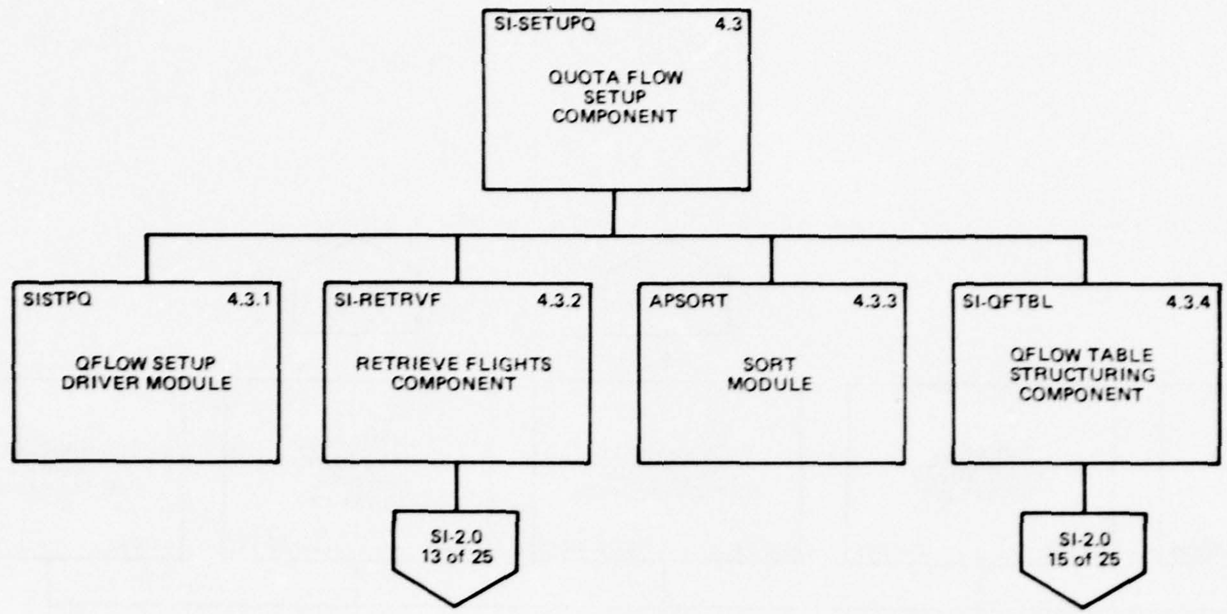


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (12 of 25)

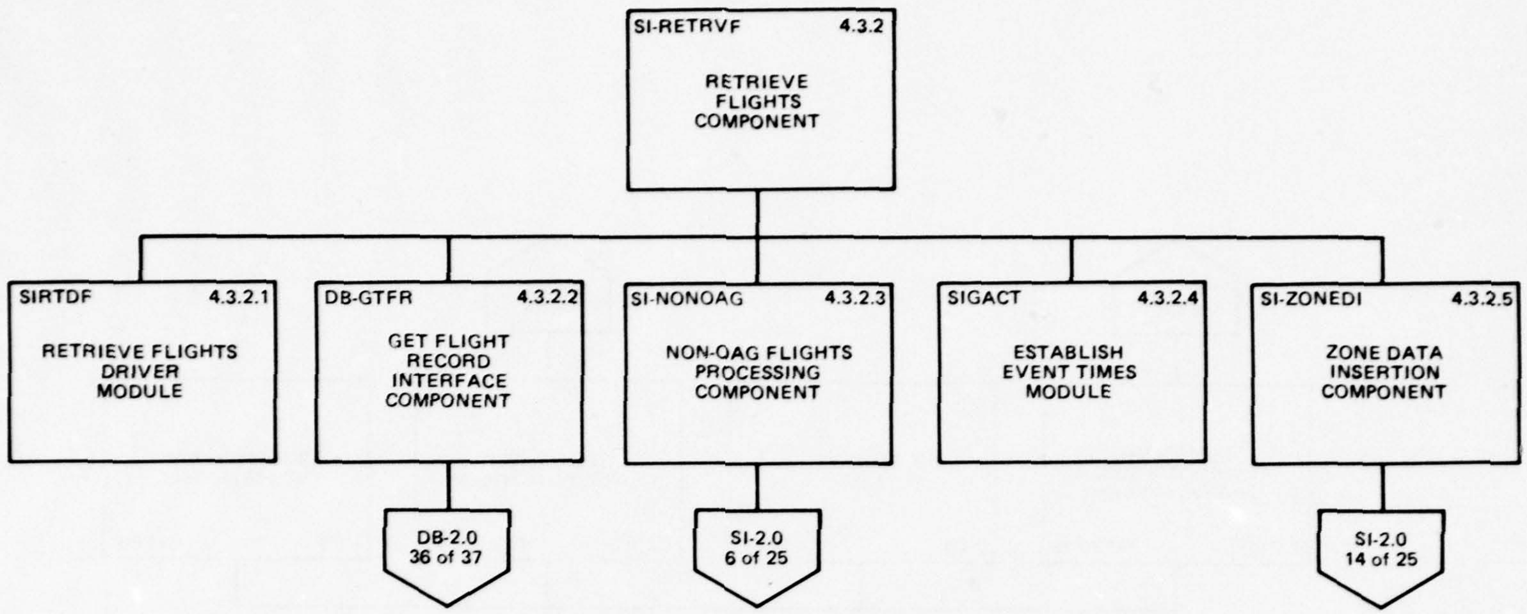


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (13 of 25)

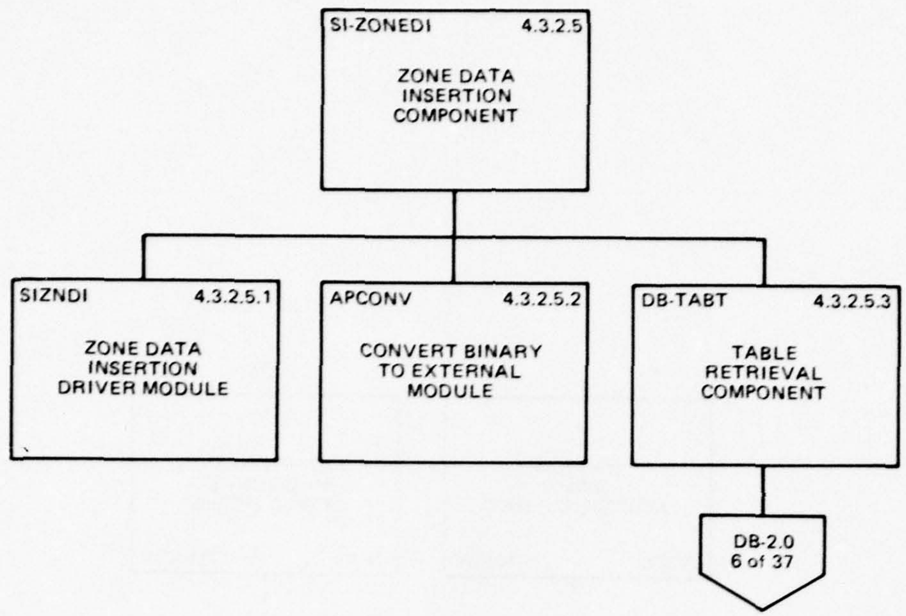


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (14 of 25)

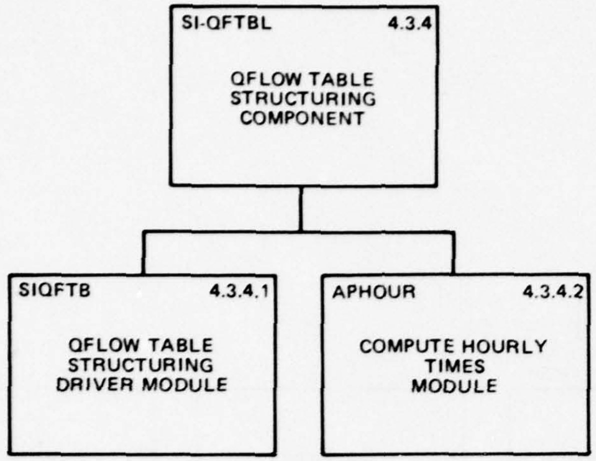


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (15 of 25)

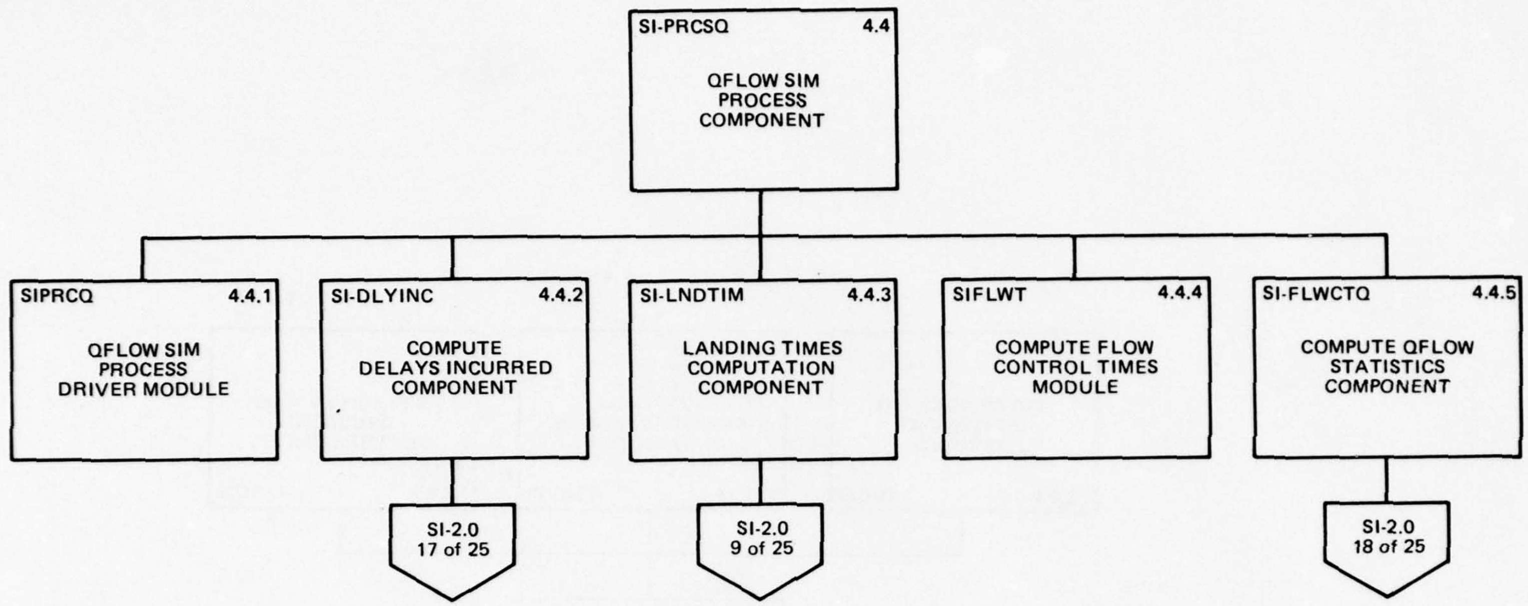


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (16 of 25)

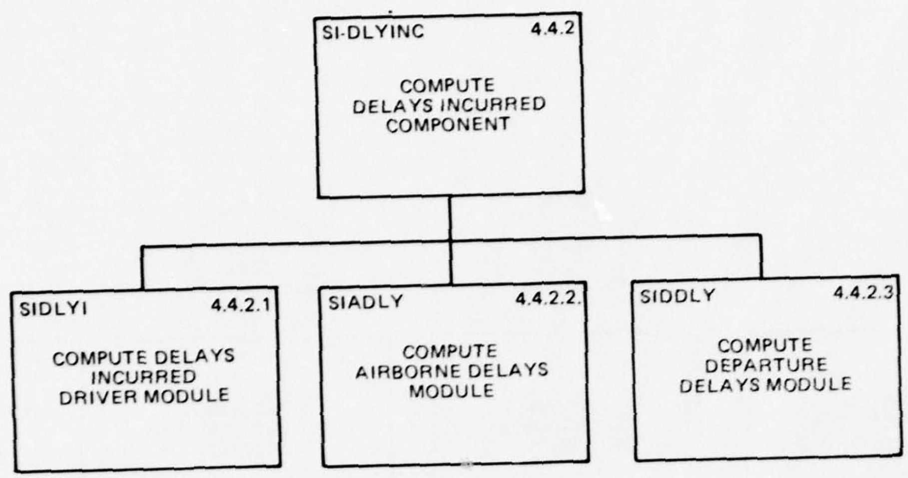


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (17 of 25)

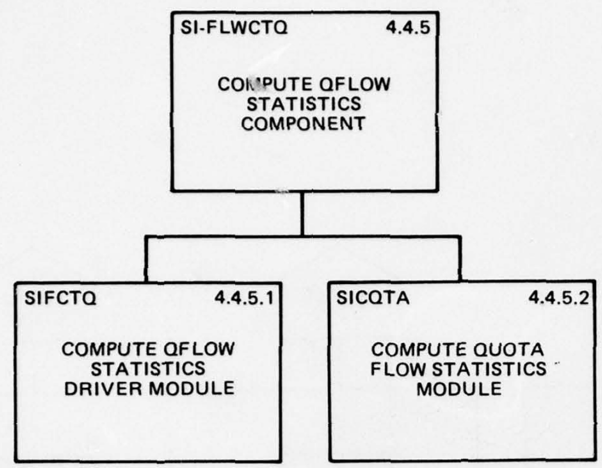


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (18 of 25)

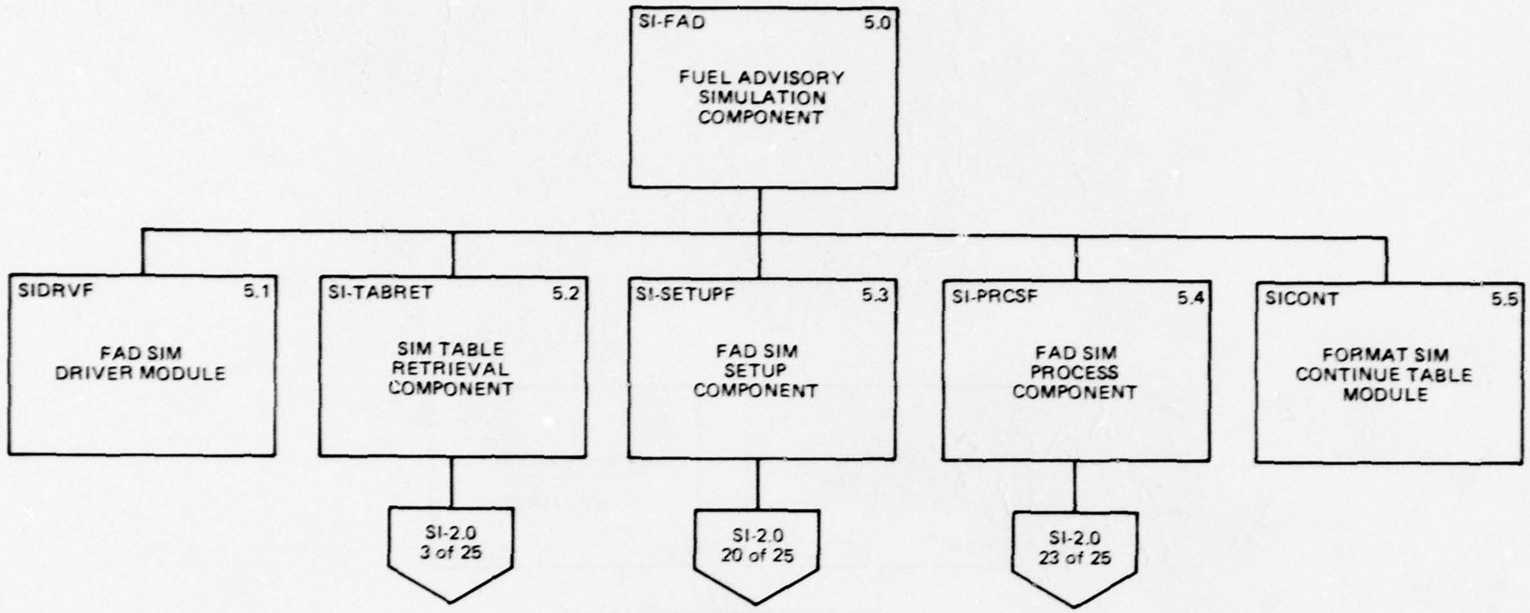


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (19 of 25)

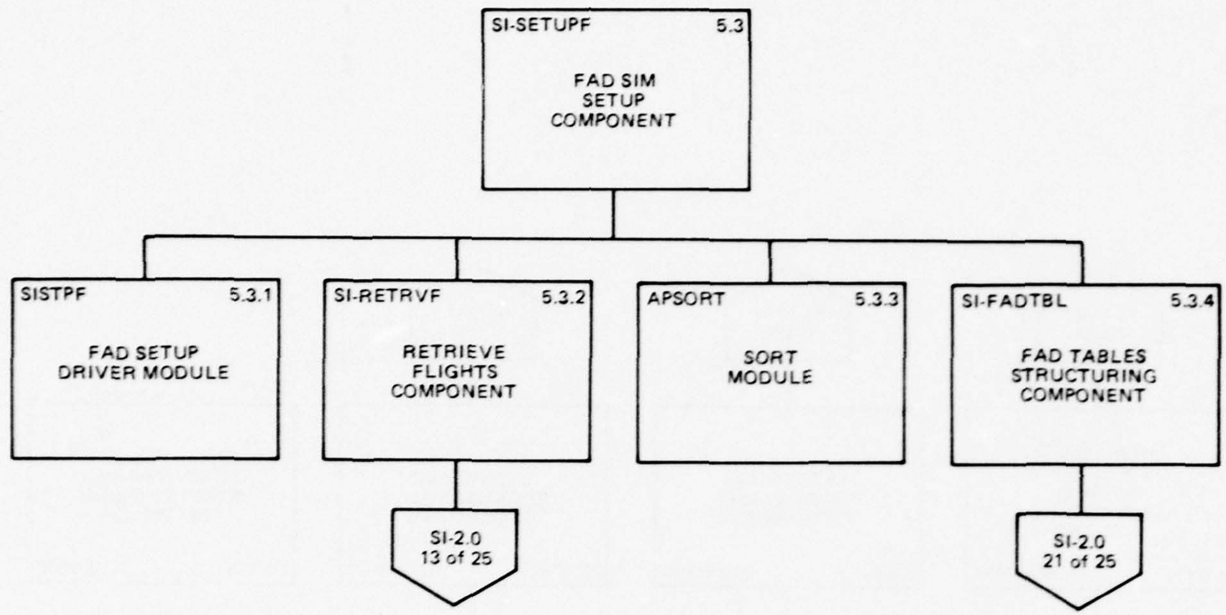


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (20 of 25)

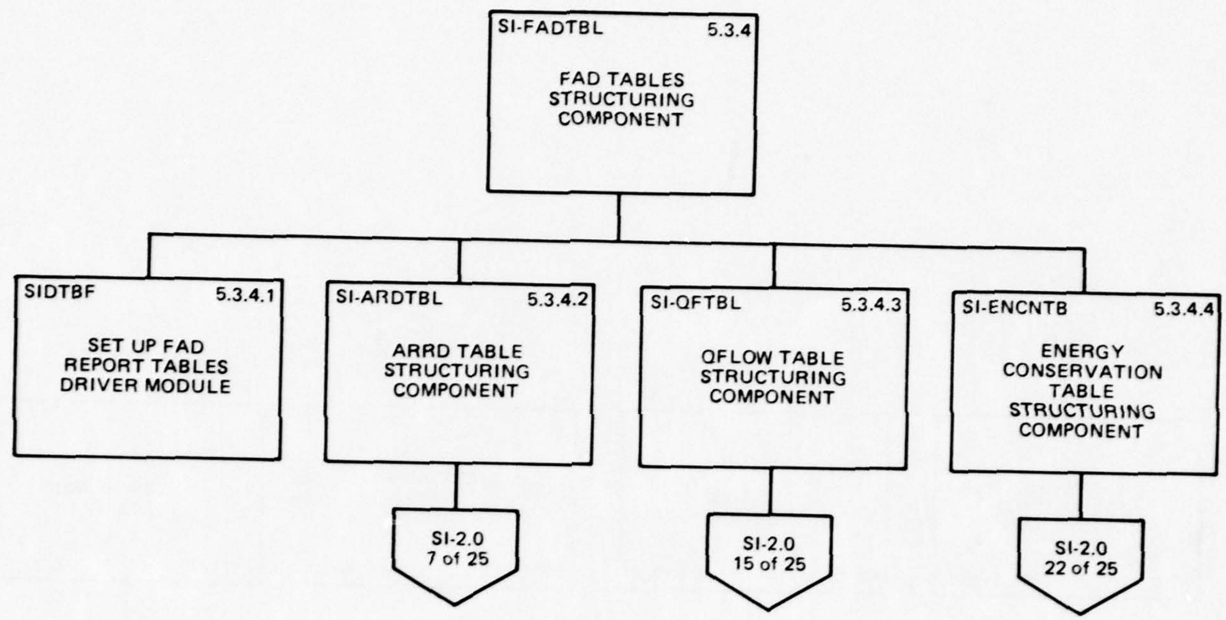


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (21 of 25)

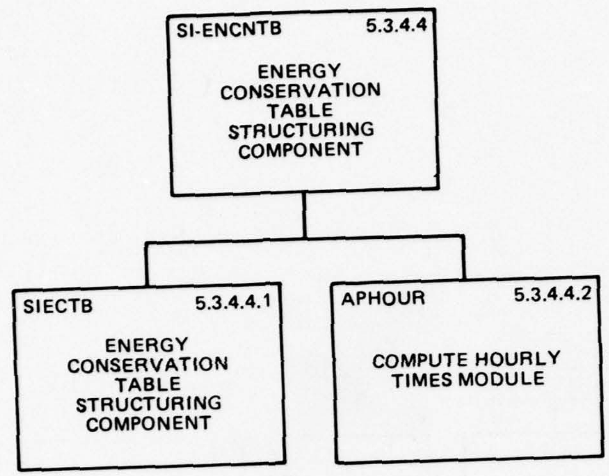


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (22 of 25)

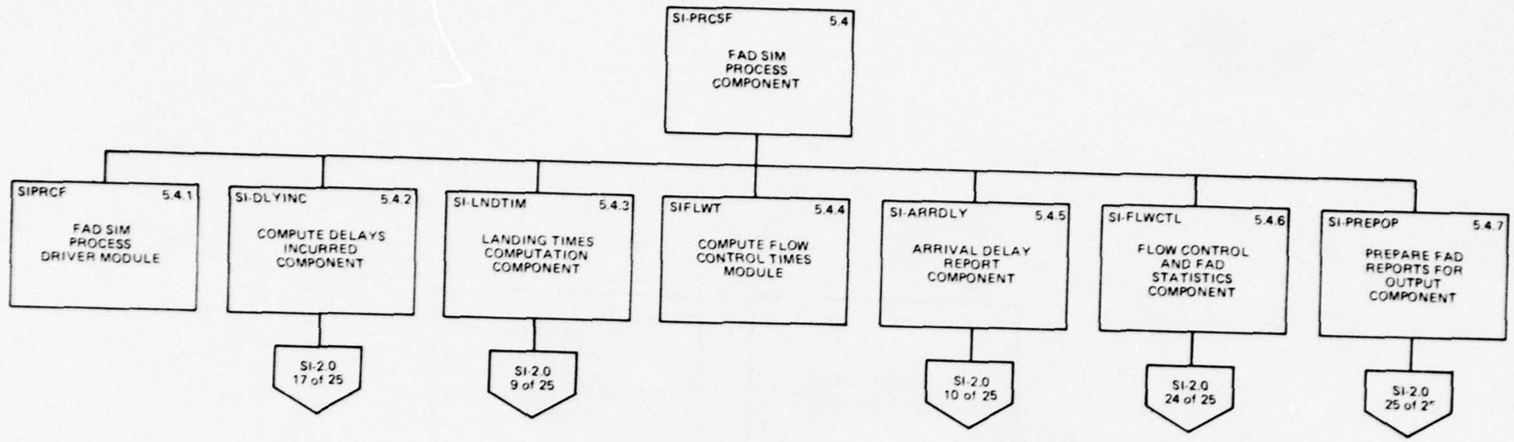


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (23 of 25)

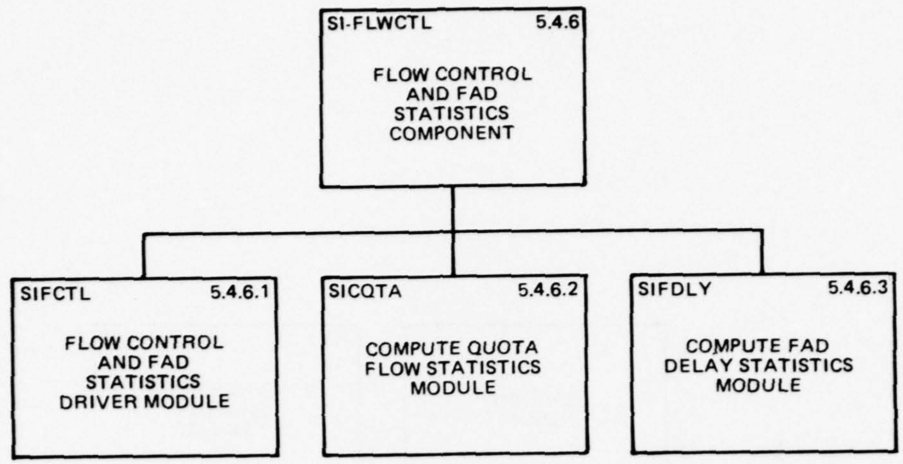


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (24 of 25)

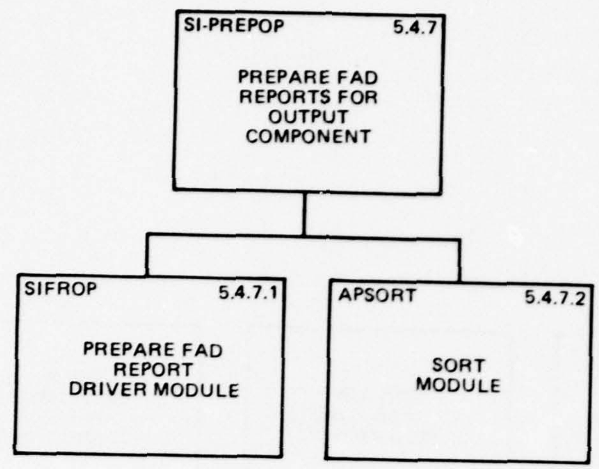


Figure 2.2.1-15. SI-2.0 Visual Table of Contents for the Simulation Subsystem (SI) (25 of 25)

2.2.1.5.4 Description Section for the Simulation Subsystem (SI)

2.0 The Simulation Subsystem (SI) performs the simulation processing specified by the input message and its qualifiers. Control is passed to the Simulation Subsystem via the Application Subsystem input component. Control is returned to the Application Subsystem update and/or output component depending on the simulation type.

3.0 The SI-ARRD Component performs the simulation processing for the Arrival Delay Predictions message. Control is passed to the component via the Application Subsystem input component. Control is returned to the Application Subsystem output component.

3.1 The SIDRVA Module controls the processing for the ARRD simulation. The module calls the Simulation Table Retrieval Component, the ARRD Simulation Setup Component, and the ARRD Simulation Process Component. If an error return occurs, the appropriate message is output and the indicated action, if any, is taken.

3.2 The SI-TABRET Component performs the retrieval of tables needed by the Simulation Subsystem.

3.2.1 The SITBTA Module sets up the calling sequence, makes the calls to the Data Base Interface for table retrievals, and processes error return codes for invalid retrievals.

3.2.2 The DB-TABT Interface Component is invoked to retrieve tables from the central data base via the Data Base Management Subsystem.

3.3 The SI-SETUPA Component performs the preparation of the ARRD Simulation flight records and report tables for processing.

3.3.1 The SISTPA Module controls the setting up of the ARRD Simulation tables and flight records. The module calls the Flight's Retrieval Component, the Sort Module and the ARRD Table Structuring Component.

3.3.2 The SI-RETRVA Component performs the retrieval of flights from the data base and the building of the ARRD simulation Flight Event List.

3.3.2.1 The SIRTDA Module controls the flight retrieval and building of the Flight Event List. The module calls the Data Base Flight Record Retrieval Component, the Non-OAG Flight's Processing Component, and the module that establishes Flight Event times.

3.3.2.2 The DB-GTFR Interface Component performs the retrieval of flight records associated with a specified airport and with scheduled arrival times within the desired limits.

3.3.2.3 The SI-NONOAG Component performs the construction of the Flight Event List using the retrieval flights and GA counts to determine the number of dummy GA events to add.

3.3.2.3.1 The SINOAG Module controls the construction of the Flight Event List. The module calls the module that establishes Flight Event times and the module that adds dummy GA flights.

3.3.2.3.2 The SIGACT Module establishes valid flight times, computes any delays which should be assigned, and validates the actual departure time field.

3.3.2.3.3 The SIGADL Module adds dummy GA flights to the Flight Event List. The number of additions is determined from hoùrly GA counts for the airport and the number of real flights retrieved.

3.3.2.4 The SIGACT Module establishes valid flight times, computes any delays which should be assigned, and validates the actual departure time field.

3.3.3 The APSORT Module orders the indexes to the Flight Event List in ascending scheduled arrival time sequence.

3.3.4 The SI-ARDTBL Component structures the ARRD report table by initializing the report hours and presetting the report fields.

3.3.4.1 The SIADTB Module controls the building of the ARRD report table. An extra report entry is created if the Stack Adjust Time is not on an hour boundary.

3.3.4.2 The APHOUR Module computes the hourly time items to be entered in the report.

3.4 The SI-PRCSA Component performs the processing of the data provided by the Setup Component. Landing times are computed for each flight, delay times are calculated, and the ARRD report is compiled.

3.4.1 The SIPRCA Module controls the processing of the ARRD Simulation. The module calls the Landing Times Computation component and the Arrival Delay Report component.

3.4.2 The SI-LNDTIM Component performs the computing of flight landing times based on the airport's hourly landing capacity and the number of aircraft arriving. Landing times are adjusted a stack adjustment and the utilization of all landing slots is maximized.

3.4.2.1 The SICFLA Module controls the computing of flight landing times. It calls the module that locates an appropriate flight when an unused landing slot is incurred. It also calls the module that resets landing times when a stack adjustment occurs.

3.4.2.2 The SICPLT Module searches the Flight Event List in order to find a flight which can utilize a computed landing slot that cannot be used by the next sequential flight.

3.4.2.3 The SIRSTK Module resets flight landing times when a stack adjustment occurs. Landing times are adjusted so that the number of flights holding at stack adjustment time is equal to the specified stack size.

3.4.3 The SI-ARRDLY Component performs the updating of arrival delay statistics and the processing of flight holding statistics. The component produces the completed ARRD report table.

3.4.3.1 The SIARDY Module controls the processing of the ARRD report table. The module calls the module that updates delay statistics and the module that processes the hold data for output.

3.4.3.2 The SIUDLY Module computes the destination hold delay for each flight and adds it to the hourly sum of delays.

3.4.3.3 The SIATOP Module computes the hourly delay averages and fills the ARRD report table for output.

4.0 The SI-QFLOW Component performs the simulation processing for the Quota Flow and Quota Flow by Zone messages. Control is passed to the component via the Application Subsystem input component. Control is returned to the Application Subsystem output component.

4.1 The SIDRVQ Module controls the processing for the QFLOW simulation messages. The module calls the Simulation Table Retrieval Component, the QFLOW Simulation Setup Component, the QFLOW Simulation Process Component, and the module which updates the Simulation Continue Table.

4.2 The SI-TABRET Component performs the retrieval of tables needed by the Simulation Subsystem.

4.3 The SI-SETUPQ Component performs the preparation of the QFLOW Simulation flight records and report tables for processing.

4.3.1 The SISTPQ Module controls the setting up of the QFLOW Simulation tables and flight records. The module calls the Flights Retrieval Component, the Sort Module, and the QFLOW Table Structuring Component.

4.3.2 The SI-RETRVF Component performs the retrieval of flights from the data base, the building of the Flight Event List, and the insertion of zone information into the Flight Event List.

4.3.2.1 The SIRTDF Module controls the flight retrieval and building the Flight Event List. The module calls the Data Base Flight Record Retrieval Component, the Non-OAG Flights Processing Component, the module that establishes Flight Event times, and the Zone Data Insertion Component.

4.3.2.2 The DB-GTFR Interface Component performs the retrieval of flight records associated with a specified airport and with scheduled arrivals times within the desired limits.

4.3.2.3 The SI-NONOAG Component performs the construction of the Flight Event List using the retrieved flights and GA counts to determine the number of dummy GA events to add.

4.3.2.4 The SIGACT Module establishes valid flight times, computes any delays which should be assigned, and validates the actual departure time field.

4.3.2.5 The SI-ZONEDI Component performs the insertion of zone data into the flight records.

4.3.2.5.1 The SIZNDI Module controls the insertion of zone data into the flight records. The module calls the binary to external format module and the Data Base Table Retrieval Component to retrieve the Zone Table.

4.3.2.5.2 The APCONV Module converts the binary zone identifier into external form for retrieval of the zone table from the Data Base.

4.3.2.5.3 The DB-TABT Interface Component is invoked to retrieve tables from the central data base via the Data Base Management Subsystem.

4.3.3 The APSORT Module orders the indexes to the Flight Event List in ascending scheduled arrival time sequence.

4.3.4 The SI-QFTBL Component performs the structuring of the QFLOW tables.

4.3.4.1 The SIQFTB Module controls the structuring of the QFLOW tables. The module initializes the

report data fields and calls the module that creates the hour entries for the report.

4.3.4.2 The APHOUR Module computes the hourly time items to be entered in the report.

4.4 The SI-PRCSQ Component performs the processing of the data provided by the Setup Component. Previously applied flight delays are removed when possible, landing times are computed for each flight, flow delays are computed, and Quota flow statistics are reported.

4.4.1 The SIPRCQ Module controls the processing of the Quota Flow Simulation. The module calls the Delays Incurred Computation Component, the Landing Times Computation Component, the module that computes flow control times, and the QFLOW Statistics Computation Component.

4.4.2 The SI-DLYINC Component performs the removal of as much previously imposed delay as possible. Any delays not removed or imposed by the departure airport remain with the flight.

4.4.2.1 The SIDLYI Module controls the computing of delays incurred by each flight. It calls the module that computes airborne delays and the module that determines what ground delays are in effect.

4.4.2.2 The SIADLY Module removes as much of the flights' previously imposed airborne or holding delays as possible.

4.4.2.3 The SIDDLY Module removes previously imposed ground delays if possible. If there are delays associated with the departure airport, the affected flights' departure and arrival times will be modified.

4.4.3 The SI-LNDTIM Component performs the computing of flight landing times based on the airport's hourly landing capacity and the number of aircraft arriving. Landing times are adjusted a Stack Adjustment and the utilization of all landing slots is maximized.

4.4.4 The SIFLWT Module computes flow control times for the appropriate flights.

4.4.5 The SI-FLWCTQ Component performs the computation of the Quota Flow statistics and fills the QFLOW table with the data to be output in the report.

4.4.5.1 The SIFCTQ Module controls the processing of the Quota Flow Statistics. The module computes release rates and carryover counts, and calls the module that computes QFLW and QFLZ simulation types.

4.4.5.2 The SICQTA Module computes the statistics for QFLW and QFLZ types of simulations. The QFLOW report will be produced for output.

4.5 The SICONT Module stores information in the Simulation Continue Table which may be used in subsequent simulations. The information includes zone and sub-zone data, hourly delay values, simulation type and report start and stop time from the current simulation.

5.0 The SI-FAD Component performs the simulation processing for the Fuel Advisory Departure messages. Control is passed to the component. Control is returned to the Application Subsystem output component.

5.1 The SIDRUF Module controls the processing for the FAD simulation messages. The module calls the Simulation Table Retrieval Component, the FAD Simulation Setup Component, the FAD Simulation Process Component, the FAD Simulation Process Component, the module that updates the Simulation Continue Tables.

5.2 The SI-TABRET Component performs the retrieval of tables needed by the Simulation Subsystem.

5.3 The SI-SETUPF Component performs the preparation of the FAD Simulation flight records and various report tables for processing.

5.3.1 The SISTPF Module controls the setting up of the ARRD, QFLOW, and Energy Conservation tables and flight records. The module calls the Flights Retrieval Component, the Sort Module, and the component that structures the various tables used in a FAD simulation.

5.3.2 The SI-RETRVF Component performs the retrieval of flights from the data base, the building of the Flight Event List, and the insertion of zone information into the Flight Event List.

5.3.3 The APSORT Module orders the indexes to the Flight Event List in ascending scheduled arrival time sequence.

5.3.4 The SI-FADTBL Component performs the structuring of the various FAD tables.

5.3.4.1 The SIDTBF Module controls the structuring of FAD tables. Calls are made to the ARRD Table Structuring Component, the QFLOW Table Structuring Component and the Energy Conservation Table Structuring Component.

5.3.4.2 The SI-ARDTBL Component structures the ARRD report table by initializing the report hours and presetting the report fields.

5.3.4.3 The SI-QFTBL Component performs the structuring of the QFLOW tables.

5.3.4.4 The SI-ENCTB Component performs the structuring of the Energy Conservation Table.

5.3.4.4.1 The SIECTB Module controls the structuring of the Energy Conservation Table. The module initializes the report data fields and calls the module that creates the hour entries for the report.

5.3.4.4.2 The APHOUR Module computes the hourly time items to be entered in the report.

5.4 The SI-PRCSF Component performs the processing of the data provided by the Setup Component. Previously applied flight delays are removed when possible, landing times are computed for each flight, flow delays are computed, the ARRD report is compiled, flow control and FAD statistics are computed, and FAD reports are prepared for output.

5.4.1 The SIPRCF Module controls the processing function of the FAD simulations. The module calls the Compute Delays Incurred Component, the Flight Landing Times Computation Component, the Compute Flow Control Times Module, the ARRD Report Component, the Flow Control and FAD Statistics Component, and the FAD Report Preparation Component.

5.4.2 The SI-DLYINC Component performs the removal of as much previously imposed delay as possible. Any delays not removed or imposed by the departure airport remain with the flight.

5.4.3 The SI-LNDTIM Component performs the computing of flight landing times based on the airport's hourly landing capacity and the number of aircraft arriving. Landing times are adjusted a stack adjustment and the utilization of all landing slots is maximized.

5.4.4 The SIFLWT Module computes flow control times for the appropriate flights.

5.4.5 The SI-ARRDLY Component performs the updating of arrival delay statistics and the processing of flight holding statistics. The component produces the completed ARRD report table.

5.4.6 The SI-FLWCTL Component performs the processing of the Quota Flow and FAD delay statistics.

5.4.6.1 The SIFCTL Module controls the processing of the Quota Flow and FAD delay statistics. The module calls the Quota Flow statistics computation module if a zone was input in the message and calls the FAD delay statistics computation module.

5.4.6.2 The SICQTA Module computes the statistics for QFLW and QFLZ types of simulations. The QFLOW report will be produced for output.

5.4.6.3 The SIFDLY Module computes the FAD delay statistics for the appropriate report intervals.

5.4.7 The SI-PREPOP Component performs the preparation of the FAD reports for output and marks the appropriate flights for update.

5.4.7.1 The SIFROP Module controls the preparation of the FAD reports for output and calls the sort module to order the flights for output.

5.4.7.2 The APSORT Module orders the indexes to the Flight Event List in ascending scheduled arrival time sequence.

5.5 The SICONT Module stores information in the Simulation Continue Table which may be used in subsequent simulations. The information includes zone and sub-zone data, hourly delay values, simulation type and report start and stop time from the current simulation.

*U.S. GOVERNMENT PRINTING OFFICE : 1979 0-627-360/1803